DESIGN EXPLORATION:

ENGAGING A LARGER USER POPULATION

A Dissertation

by

JOHN MICHAEL MOORE

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2007

Major Subject: Computer Science

DESIGN EXPLORATION:

ENGAGING A LARGER USER POPULATION

A Dissertation

by

JOHN MICHAEL MOORE

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved:

Chair of Committee,	Frank M. Shipman, III
Committee Members,	Richard K. Furuta
	William Lively
	Steven Smith
Head of Department,	Valerie Taylor

August 2007

Major Subject: Computer Science

ABSTRACT

Design Exploration: Engaging a Larger User Population. (August 2007)

John Michael Moore, B.S., Texas A&M University;

M.S., Southwest Texas State University

Chair of Advisory Committee: Dr. Frank M. Shipman, III

Software designers must understand the domain, work practices, and user expectations before determining requirements or generating initial design mock-ups. Users and other stakeholders are a valuable source of information leading to that understanding. Much work has focused on design approaches that include users in the software development process. These approaches vary from surveys and questionnaires that garner responses from a population of potential users to participatory design processes where representative users are included in the design/development team. The Design Exploration approach retains the remote and asynchronous communication of surveys while making expression of feedback easier by providing users alternatives to textual communication for their suggestions and tacit understanding of the domain. To do this, visual and textual modes of expression are combined to facilitate communication from users to designers while allowing a broad user audience to contribute to software design. One challenge to such an approach is how software designers make use of the potentially overwhelming combination of text, graphics, and other content.

The Design Exploration process provides users and other stakeholders the Design Exploration Builder, a construction kit where they create annotated partial designs. The Design Exploration Analyzer is an exploration tool that allows software designers to consolidate and explore partial designs. The Analyzer looks for patterns based on textual analysis of annotations and spatial analysis of graphical designs, to help identify interesting examples and patterns within the collection. Then software designers can use this tool to search and browse within the exploration set in order to better understand the task domain, user expectations and work practices. Evaluation of the tools has shown that users will often work to overcome expression constraints to convey information. Moreover, the mode of expression influences the types of information garnered. Furthermore, including more users results in greater coverage of the information gathered. These results provide evidence that Design Exploration is an approach that collects software and domain information from a large group of users that lies somewhere between questionnaires and face to face methods.

DEDICATION

To my family

ACKNOWLEDGEMENTS

There are many people I should name while making my acknowledgements. However, I'm sure that I would miss someone. Also, my allotted space for acknowledgements would not be sufficient to name all of the people who have played a part in my doctoral studies here at Texas A&M, regardless of how small. So, I will highlight some of the most influential and crucial individuals who made my dissertation possible whether it was through academic or moral support.

First I want to thank my parents. They have always supported me in my endeavors, even though they don't always really know what it is that I do. Here I also want to recognize my sister and brother. My brother managed to finish his bachelor's degree here at Texas A&M just before I completed my dissertation.

Next, I want to recognize my committee. First I want to thank my advisor, Dr. Frank M. Shipman III. He has been helpful and patient as my dissertation topic evolved from a classroom project into a funded grant project. Without his feedback, advice, guidance, mentoring, and especially his nudging at the end, I'm not sure when I would have finished. In addition to providing feedback in framing my work, Dr. Steven Smith was also crucial in helping me procure the subjects for my first study. I also want to thank Dr. Richard K. Furuta and Dr. William "Mac" Lively for their time and feedback.

My friends and colleagues in the Center for the Study of Digital Libraries have been an invaluable resource. I want to thank Haowei, Luis and Unmil. I'll never forget the times when we would go through our presentations to ensure that they were the best

that they could be. Moreover, Haowei was an excellent sounding board for working through ideas and has become a great friend. Unmil provided many alternative perspectives about people and life in addition to scholarly pursuits. I should also extend my gratitude to Kushal who was my partner for the class project that eventually became my dissertation topic.

Texas A&M is an environment for learning more than just academics. Its student organizations provide leadership experience and exposure to a diverse population. I want to thank those friends I made while working with the GSC (Graduate Student Council), the CSGSA (Computer Science Graduate Student Association) and other student organizations as well.

I made many friends during my time at Texas A&M, and I want to recognize some of the closest ones. Tamer and Ayman, my friends from Egypt, offered me friendship that gave me an escape from academics through most of my years. Ayman was a great support during my final push to finish. I also want to express thanks to Yakut. Seeing her finish helped spur me to completion. Tamra also provided friendship, conversation and escape from academic rigors. Lastly, I want to give my thanks to Darren, who has become one of my regular friends for eating out and having conversations about almost anything except school.

I don’t want to neglect my good friends outside of Texas A&M. I want to thank Colleen and Heather, the twins who were never afraid to tell it to me like it is while still offering their love and friendship.

Finally I want to thank those in the PhD support group, especially Brian who supervises the group. The time I spent in the group helped me learn to prioritize and make progress on my dissertation work, even at times when I didn't think that progress was possible.

I know that there are many people that I've left out, but as I start thinking of another name it leads to even more names that I could mention and space is limited. Thanks to everyone who has touched and impacted my life while I've been at Texas A&M.

This work was supported in part by NSF grant 04-38887.

TABLE OF CONTENTS

Page

Page

Page

Page

LIST OF FIGURES

Page

Page

Page

LIST OF TABLES

Page

1. INTRODUCTION

When creating interactive software, designers must understand the domain, work practices, and user expectations before determining formal requirements or generating initial design mock-ups. Users and other stakeholders are a valuable source of information leading to that understanding. The early phases of software development are an ideal time to get input from a large segment of users. During this time, requirements information is gathered for analysis prior to the formation of any formal requirement specifications or initial designs. Gathered information can be integrated into the development process much more cheaply here than at later stages.

Much work has focused on design approaches that include users in the software development process. These approaches vary from surveys and questionnaires that garner responses from a population of potential users to participatory design processes where representative users are included as members of the design/development team. Questionnaires and surveys reach a large number of users but have a low rate of return and often elicit limited details when they are returned. On the other hand, participatory design generates rich feedback but the number of users is limited to a few representative users due to time limitations and costs.

This dissertation investigates an approach that lies between these extremes called “Design Exploration”. The goal is to provide a communication medium to elicit more

---

This dissertation follows the style of the *ACM Transactions on Information Systems.*

information than surveys while maintaining a low-overhead per participating user. While not meant to generate the rich information provided via participatory design or other methods requiring face-to-face meetings, it can be used to validate and enhance feedback from such a process. Addressing this middle ground provides a technique that broadens the number of users that can contribute to software design.

The effectiveness of the various approaches to including users relies on the success of communication. In fact, it is often failed communication that leads to inadequate requirements specifications [Potts et al. 1994]. This is often exacerbated for communication that does not occur in face-to-face situations. While face-to-face communication does allow for the repair of communication breakdowns [Suchman 1987], the role users see themselves taking in this interaction can alter the expression and elicitation of design information [Boland 1978].

This approach retains the remote and asynchronous communication of surveys while making authoring of feedback easier by providing users alternatives to textual communication for expressing their suggestions and tacit understanding of the domain. Combining visual and textual modes of expression facilitates communication from users to designers. Providing this communication in reference to an artifact can facilitate expression by allowing "design by doing" [Ehn 1988]. Glenberg and McDaniel [1992] have noted that integrating spatial and linguistic information is a requirement for effective communication.

The Design Exploration process provides users and other stakeholders a construction kit where they create annotated partial designs consisting of windows,

widgets and textual explanations. The volume of expressions produced by users can become quite large. Manually analyzing each user's set of windows, widgets and annotations would be time consuming and costly. Clearly, assistance in the analysis phase of this process is vital for this method to be successful. To address this need an exploration tool allows software designers to search and browse the collection of partial designs in order to better understand the task domain, user expectations and work practices. Searching and browsing is supported by textual analysis of annotations and spatial analysis of graphical designs. These help identify interesting examples and trends within the collection.

To explore the potential of the Design Exploration process, two tools were developed. The Design Exploration Builder allows end users to construct annotated partial designs to express their understanding of the software's use. The Design Exploration Analyzer provides a software designer access to a set of annotated partial designs and the ability to navigate between designs based on textual and spatial analysis of the information provided by users. The benefits and difficulties associated with each of these two tools were evaluated via human subject experiments.

Section 2 overviews prior work on including users in software design and requirements gathering. Section 3 presents the Design Exploration approach in the abstract while Sections 4 and 5 describe the Design Exploration Builder and Design Exploration Analyzer respectively. Section 6 presents the study of the Design Exploration Builder, and Section 7 presents the study of the Design Exploration

Analyzer. Section 8 describes open issues and future work. Conclusions of this investigation of the Design Exploration approach are in Section 9.

2. INCLUDING USERS IN SOFTWARE DEVELOPMENT

Although common in the past, it is now rare for interactive software to be developed without involving users in some way. It is relatively easy to involve a small number of users in a meaningful way; however, as the number of user participants increases, it becomes more difficult for user input to occur in more than a cursory manner.

For any software development effort, some type of requirements collection and analysis is necessary to define the system. Since errors from this phase can profoundly affect later phases of software development and quality of the final product, it is also viewed as the most critical step [Dardenne et al. 1991]. Further, the measure of a system's quality depends on how well what is built matches the requirements [Finkelstein 1994]. Frequently, the documented requirements become the template for developing and then evaluating the success or failure of the final system. However, not all requirements (i.e. expectations) that users will employ to evaluate a system have been identified, documented or allowed to emerge. Users may also have unstated or implicit expectations that they mistakenly believe are fulfilled by other requirements. These undocumented requirements can stem from tacit knowledge, i.e. knowledge that users do not know that they have or are unable to express [Polanyi 1966]. Regardless, these undocumented requirements are used to measure the success or failure of the final system and might be more important than documented requirements.

Stakeholders, those who have a vested interest in a project, are the information source for requirements. It becomes vital that the requirements be specified and analyzed

in sufficient detail so that they more completely match the expectations of all stakeholders or that requirements are allowed to emerge through the development process. Getting a stable set of user requirements is further complicated by the fact that requirements are rarely static and continue to change throughout the software development process [Curtis et al. 1988]. Furthermore, requirements will appear to fluctuate when based on an incomplete requirements analysis. Various requirements elicitation approaches provide different ways that help users express their implicit requirements and unlock their tacit knowledge. Regardless, capturing more information early can help stakeholders converge on a more stable set of requirements, thus reducing the number of requirements changes needed throughout the software development process.

2.1 Communication

Effective communication is crucial for a successful system, especially when determining requirements [Holtzblatt and Beyer 1995]. Consequently, it is often failed communication that is behind inadequate requirements specifications that lead to the failure of a system [Potts et al. 1994]. Communication between stakeholders, including users and developers, must convey all the information necessary to create a software system.

Methods for eliciting requirements and developing software occur in a particular communication context that influences the outcomes of resulting information exchanges. Factors affecting context include whether users and developers are together or are in different locations, whether interactions are synchronous or asynchronous, when in

software development the communication occurs, the number of individuals involved in the communication, the role of users, and the modality of communication (e.g. textual or graphical). Naturally, there are trade offs so no single technique can utilize the strengths of each factor.

2.1.1 Specificity

User expectations range from high-level functional requirements to fine-grained procedural expectations that describe how the system will behave. It is possible to fulfill a set of functional requirements in multiple ways using different instantiations of procedural requirements or behaviors.

The initial requirements are often a set of broad ranging goals stated at conception. Frequently, these conceptual requirements motivate the development of a given software package. Those initiating software development generally know the basic tasks that need to be performed as well as their ideas about how the software should "feel" when used. The requirements, specified at this level, are functional requirements that give little indication regarding the procedural requirements that are entailed. The procedural requirements comprise the set of processes that need to be automated and integrated with the practices of end users. Getting input from end users about procedural requirements can be just as important as the functional requirements.

2.1.2 User Role

End users take on different roles within the context of approaches that collect software expectations. Some approaches provide needed information but place the user in the role of informant rather than participant [Muller et al. 1993]. In such situations the

software developer is the expert and end users provide information for them. Users can also be seen as experts while acting as an information source. Recognizing the primacy of users in interactive software systems, there has been a push to place users on equal footing with software developers. While some approaches lend themselves better to one perspective or the other, users' perception of their role can still be influenced by how they are introduced to the interaction [Boland 1978].

2.1.2.1 Participatory Design

Many software development processes have been shifting towards participatory design (PD), where users take a more active role [Carroll et al. 1997]. PD, also known as Scandinavian Design, is defined by two features [Ehn 1993]. The first is democracy, power and control in the workplace. "People who are affected by a decision or event should have an opportunity to influence it." [Schuler and Namioka 1993] The second is the inclusion of skilled users in the design of software under the premise that their participation can improve quality.

One major advantage of PD is that participants form a personal stake in the product and are more likely to work to make it succeed. It helps develop a sense of ownership within the user community. As with interviews and task analyses, PD requires user time, can be difficult to schedule and can be costly.

PD can occur at different points in software design, including testing, requirements elicitation and any other activity that involves users and gives them power during software development [Muller et al. 1993].

2.1.3 Communication Modalities

People communicate through various modalities. Frequently, this communication is verbal or textual. Even when using verbal expression, communication is enhanced and even altered by visual cues such as facial expression, body language and gestures. Moreover, auditory inflections and intonations can alter the meaning of the textual content conveyed. Another mode manifests as graphical and spatial expression. Sketches and drawings provide avenues of communication that might be difficult or impossible to convey using text alone. Graphical constructs alone do not always communicate what they mean until they are learned. Meanings of signs, even those that have a strong iconic component, are often not understood a priori. So they must first be explained to be understood. Text can reduce such ambiguity found with graphical communication and allow viewers to learn the meaning of graphical expressions.

People have various styles of communication utilizing different modalities. For example, some people give verbal driving directions while others draw maps and still others write out text. Moreover, some modalities are better suited for conveying certain types of information. It is difficult for people to textually describe practices that they do not normally describe, and if they do, their descriptions are unlikely to accurately represent the practice [Goguen and Linde 1993]. So, constraining people to any one mode of communication will inhibit the successful expression of information across a diverse population. When users can take advantage of their preferred modes of expression or modes that more closely match what they are trying to describe, they can more fully convey information.

2.1.4 Locality and Temporality

Much of the ambiguity encountered through a written mode of communication can be resolved through synchronous interactions. Through these interactions a rich set of information is available for use and interpretation. On top of the language itself, facial expression, body language and intonation provide depth of meaning to the language. Also, the process whereby individuals check for understanding and repair breakdowns in communication occurs more quickly in face-to-face meetings than outside of that context [Suchman 1987]. Face-to-face communication has many benefits. However, these interactions may limit the exchange of some information due to the influence of the software designer on the end user and the end user's role in the interaction. The end user can be guided down a specific path that matches a particular interest or expectation of the software designer.

In today's global community, stakeholders are more and more likely to be spread across geographically disparate areas. The most effective techniques require face-to-face interactions (synchronous in time and location). These are expensive in time and often require co-location. Technology for supporting synchronous communication for individuals in different locations can range from a telephone, to video conferencing to chat programs online. Although technology for facilitating synchronous interactions are continually improving, many of the affordances of face-to-face interactions are lost or poorly imitated and might even increase the time needed for successful communication. While these technologies can sometimes bypass certain social protocols, it is unclear whether this negatively or positively influences the interaction. For these reasons, crucial

interactions between end-users and software developers may be limited, given only “lip service” to fulfill a contractual obligation or not even happen. This happens in spite of the fact that approximately half of software project failures in the United States and Europe are attributed to requirements problems [Lamsweerde 2000].

2.2 Approaches

One example of a tool that works to reduce the time required for requirements gathering is the Requirements Apprentice (RA) [Reubenstein and Waters 1991]. RA develops a coherent internal representation of a requirement from an initial set of disorganized, imprecise statements. The initial set of data input by the software engineer is based on interviews with end users. RA does not interact directly with end users to avoid “the syntactic complexity of natural language input.” Although RA works to provide a better set of initial requirements, it still relies on other methods for initial requirements acquisition.

Another tool that assists requirements elicitation is KAOS [Dardenne et al. 1991]. KAOS supports goal-directed concept acquisition which focuses on higher level goals rather than more detailed procedural requirements. The authors view elicitation with this tool “as a cooperative learning task between clients and analysts.” This is because requirements acquisition is driven by a formal model that clients must learn.

These as well as other techniques are available for elicitation and refinement of requirements [Goguen and Linde 1993; Nuseibeh and Easterbrook 2000]. These approaches frequently focus on the representation of requirements information and techniques to elicit the information needed for that representation. While getting formal

representations of requirements can be important, the constraints of formal representations can negatively affect what information is collected [Shipman and Marshall 1999].

2.2.1 Interviews and Questionnaires

Questionnaires/surveys and interviews are traditional techniques for acquiring information from users [Nuseibeh and Easterbrook 2000]. Goguen and Linde [1993] categorize all of the above as types of interviews. According to this perspective, a questionnaire is an interview without synchronous face-to-face interaction.

Questionnaires can include questions with varying levels of specificity. Specific questions include focused short answer, multiple choice questions and attitudinal measures (e.g. Likert scale). While they tend to make answers easier to collate for interpretation, such questions can bring assumptions about the domain or design to users' awareness. They also limit responses to those provided which further enforces assumptions written into the question. More open-ended questions are less likely to lead respondents in a particular direction (whether intentionally or not), but interpreting results presents a challenge. Open-ended questions such as "What should this tool do for you?" will result in a list of features but not how these features are interrelated [Moore and Shipman 2000]. While users know the functions that the software should include, they do not explicate the fine-grained procedural behavior that they desire. One reason for this omission is that they may believe that the high-level functional descriptions of their tasks entail the detailed procedural steps and processes that are not made explicitly.

Regardless of a question's specificity, the words printed on the questionnaire will be interpreted differently by each respondent [Suchman and Jordan 1990] further hindering successful communication. Additionally, individuals feel they are communicating with another person when responding to a questionnaire. This communication assumes a shared background knowledge that supports the exchange of information. However, when attempting to elicit detailed procedural information, these assumptions can inhibit the successful exchange of ideas. The user may not verbalize pertinent information with the assumption that the person reading the questionnaire already knows these things. So, getting useful and necessary information from users is inherently difficult and compounded by the inexact nature of language.

Survey interviews, open ended interviews and focus groups can be used as an alternative to written questionnaires. Survey interviews are similar to questionnaires but where answers are collected by an interviewer rather than through written responses. Open ended interviews are more exploratory, but do not lend themselves to the types of quantitative data that can be collected in questionnaires and surveys. Focus groups can be seen as an open ended interview with a group of people. In all of these cases the interviewer can take advantage of the affordances of face-to-face interaction to get more meaningful results from users. Where these interviews are done on the phone, visual cues are lost, but affordances such as tone, volume, clarification and communication repair can still occur.

2.2.2 Ethnography and Task Analysis

Ethnographic approaches collect information by observing what workers do. These observations can identify how tasks are actually performed in the real world verses idealized or rationalized versions given when people asked how tasks are performed. End users may not even be aware of practices that observers identify. Ethnographic analysis gives a software designer access to many of the rich features of human communication mentioned above. Unfortunately, people may break from the routine that the observer is trying to capture when they know or are reminded that they are being observed. As a result discussion of work practices frequently does not occur until follow-up interviews.

The think-aloud technique tries to get users to communicate what they are thinking as they perform a task. This might not be valuable for pure task analysis since the talking interferes with the performance of tasks and descriptions probably will not match the reality as noted earlier. However, it does potentially provide access to useful information when the process is better understood [Nielsen et al. 2002].

One approach that combines ethnography and interviews is contextual inquiry. In this approach the interviewer follows a user at work [Beyer and Holtzblatt 1999]. As the user performs tasks, the interviewer asks questions to understand the actions. After the interview, a team (which may not include the user observed) interprets the information gathered. Holtzblatt and Beyer [2003] have developed InContext, a tool to assist this process.

Viller and Sommerville [1999] present modifications of ethnography to integrate it into the requirements engineering process. These modifications address issues of time, modeling of data, and process to transform it into a more formal method for requirements engineering.

Task analysis is the primary focus for design activities [Wilson and Johnson 1995] in task-based design. ADEPT is a software tool to support this approach by modeling tasks and tracking requirements. Moreover, users participate in the task analysis process.

Ethnography and task analysis provide information about current work practices. However, too much focus on current practices can blind the identification and development of transcendent practices. Regardless, all of these practices require a lot of time per user observed/included and thus tend to include few users.

2.2.3 Prototyping

Prototyping provides a way of “interacting” with systems that have not yet been built. This is beneficial because many issues can be resolved before building the final interface. Prototyping can be either low or high fidelity. High fidelity prototypes look very much like a polished interface and functionality is created to accurately simulate the feel of the application. For this reason, as users interact with the prototype issues are easy to identify. However, users might perceive this as being a finished system and they can be expensive to create. There is also danger of coding compromises made to create a prototype quickly persisting to the final system along with its failings.

Low fidelity prototypes have an obvious appearance of being incomplete. Because they are cheaper to create, multiple designs can be created. However, they are easier to change and users are more likely to suggest changes than with a more finished interface. These characteristics make it very useful for the early stages of requirements gathering [Rudd et al. 1996]. Since the interaction is not well defined they tend to be demonstrated to users rather than having users interact with the prototype. Consequently, interaction issues may not be identified.

Low fidelity prototypes, sometimes called mock ups, are frequently done on paper and therefore the resulting information is not easily transferred into computing systems. To address this issue SILK allows designers to sketch interfaces and uses sketch recognition to identify the components of the interface [Landay 1996; Landay and Myers 2001].

In many mock up techniques users respond to prototypes created by designers. However, in the spirit of participatory design, users can be the creators of mock ups. In PICTIVE [Muller 1991; Muller et al. 1993] users and designers work together using post it notes and objects acting as interface components to build an interface on a physical shared table space. Whereas PICTIVE requires physical co-location, TelePICTIVE [Miller et al. 1992] and PICTIOL [Farrell et al. 2006] moves the shared space to a collaborative computer application. While GUI developers are still involved, the tool is tailored to “naive" users so they can participate.

Prototypes can be incorporated as the focus of discussion for other requirements gathering approaches. For example Sutcliffe [1995] uses a prototype in a group

discussion format to evaluate interface alternatives and identify additional requirement issues.

2.2.4 Scenarios

"Scenarios can be thought of as stories that illustrate how a perceived system will satisfy a user's needs." [Holbrook 1990] Scenarios tend to use generic agents and actors. Consequently, when used by software developers they struggle to visualize the impact of design decisions since the consequences on users are abstract and choices can be rationalized by saying "what if." Bødker [2000] notes it is easier to apply common sense to a caricature than to something that is "middle ground." Along this same vein, personas have become a popularized mainly due to Cooper's *The Inmates are Running the Asylum* [2004]. Grudin and Pruitt [2000] make an argument that encourages persona use while still taking advantage of user participation.

Scenarios situate users to help them think through design alternatives [Kyng 1995]. Since users are not designers, Kyng suggests some kind of scaffolding to support scenario based design. Potts proposes one approach to doing this called ScenIC which is an inquiry driven approach based on characteristics of user memory [Potts 1999].

Regardless, scenarios are used for requirements elicitation [Chin et al. 1997; Holbrook 1990; Potts 1999]. Moreover, scenarios are advocated for the entire design process [Carroll 2000].

2.2.5 Summary

While a rich source of requirements these approaches where users participate can be time consuming as they require users and software designers to be co-located in time

and space. The related scheduling issues can draw out the requirements analysis process, lengthening the overall software development time-table. In some approaches, there must be more than one or even ongoing regular meetings. This further increases the overhead including scheduling.

Some level of face-to-face interactions with users is necessary in most software design. These meetings would be aided if the initial set of requirements represented a more detailed representation of user expectations. A tool that allows initial requirements gathering to occur outside the realm of face-to-face interactions while generating a richer initial set of requirements has the potential to reduce the amount of user time actually needed.

The problem remains that requirements gathering methods tend to fall into two categories: those which produce rich results but are expensive (in time and money) and those that are less expensive but also less informative.

3. APPROACH: DESIGN EXPLORATION

The approach presented here, called Design Exploration, provides a way of including users in software design that is between the extremes of collecting limited information from a large number of users and rich information from a few users. It also incorporates some of the benefits of participatory design. It is not meant as a replacement for face-to-face communication but as an alternative approach to asynchronously gather user input and to enhance the value of this input in the design process.

3.1 Communication through Design

User interfaces enable communication between humans and computers where inputs are communication from the human and outputs are communication from the computer. Successful interface designs indicate successful communication where poor designs indicate less successful or failed communication. Consequently, in interactive systems, defining the communication that crosses this human-computer interface is crucial.

In the context of a graphical user interface (GUI), this communication occurs through the use and organization of widgets in windows. The use of widgets has become standardized over time due to the prevalence of graphical operating system interfaces such as Microsoft Windows, the Macintosh OS and graphical interfaces for various flavors of Unix. Through this interaction paradigm the placement and usage of widgets essentially function as a visual language, the language of the GUI. As users become accustomed to using these widgets, they begin to implicitly understand this language.

Users see radio buttons and think “one choice” where software developers see “mutual exclusion.” This can be viewed as a “third space” since it acts as a bridge between the users’ space and the software developers’ space [Muller 2002].

Combining communication with reference to an artifact can facilitate communication by allowing “design by doing” [Ehn 1988]. Glenberg and McDaniel [1992] have noted that integrating spatial and linguistic information is a requirement for effective communication. Others have used a combination of visual and textual information. Reeves and Shipman [1992] use visual design artifacts as the focus for communication. Similarly, the Visual Knowledge Builder (VKB) uses visual objects for information organization and interpretation [Shipman et al. 2001a].

3.2 Annotated Designs

In Design Exploration, interface construction becomes one way (but not the only way) for end users to communicate their desires and goals for software. End users can convey things that might be too difficult when expression is limited to text. Conversely, limiting expression to these primarily graphical representations may also result in ambiguous expression. Interface constructions can be clarified by attaching textual argumentation to graphical artifacts, allowing the strengths of both textual and visual information to synergistically augment communication.

The combination of modes of communication is not only valuable for easing user expression. Software designers can identify how textual vocabulary, as presented by users, fits into their understanding of the domain environment based on their spatial/graphical expression and use of the language of the GUI. This is important since

the terminology used for requirements should match the environment where the system will be used [Zave and Jackson 1997]. The graphical constructions facilitate the transfer of these concepts from end-user to software designer, thereby avoiding many of the trappings of jargon and textual communication.

3.3 The Design Exploration Process

The Design Exploration approach allows users to communicate requirements information through the construction of graphical user interface mock-ups augmented with textual argumentation, i.e. descriptions and explanations [Moore and Shipman 2000; Moore and Shipman 2001; Moore 2003]. Figure 1 shows an overview of the entire process. Using a construction kit, users individually create mock-ups, built with windows and widgets and attached argumentation, as communication to software developers. The resulting annotated partial designs are collected and analyzed by software designers engaged in requirements definition and evolution.

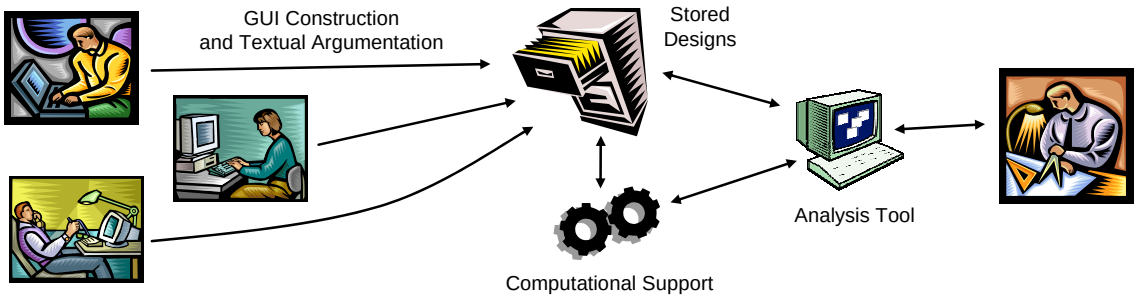


Figure 1. Design Exploration process.

A Design Exploration analysis tool assists software designers as they mine the data collected from users. End users provide graphical design ideas and textual

argumentation that are meant to provide relationships and information. This can create a potentially large data set that can be daunting for analysis by hand.

The Analysis Tool needs to support access to both the graphical and spatial expression and the textual annotation. There are a variety of algorithms for the analysis of text and for the analysis of graphical layouts [Landay and Myers 1995; Shipman et al. 1995]. Our approach combines these approaches to provide a variety of search and navigation options based on the results of textual and spatial analysis. This approach leaves the software designer in control of deciding what to see and when to see it. Additionally, combining search and browse options allows for a variety of work practices based on the software designers' understanding of the collection, their prior access to associated/related items, the focus of their current activity, and their preferences. This combination of searching and browsing is prevalent today on the Web.

3.4 Scenario

To help envision this approach in use, imagine this scenario. A software company is developing new software for inventory control since new technologies such as RFID tags are now frequently used but are not accounted for in the system. To start collecting input for the new system, employees that work with inventory are sent a software program (the Design Exploration Builder) that allows them to create mock ups of the interfaces they expect for the system. At the same time they are sent information about what RFID tags are so they can incorporate these into their designs. After the employees have created their partial designs, they email the designs to the software design team. Once a set of designs are collected, the software developers explore those

designs using the Design Exploration Analyzer to get ideas about changes in existing features and new features. Searching for “RFID” gives quick access to where RFID tags appear in the designs to give insight into how different users perceive RFID tags relate to other features in the inventory control system. As this information is explored, bits of information, requirements and questions are generated to seed further methods for requirements elicitation and design.

The Design Exploration process involves expression by potential users and stakeholders of annotated partial designs followed by access to and use of the resulting collection of artifacts by software designers. The following sections describe the two tools implemented to enable Design Exploration. The Design Exploration Builder allows potential end users to create a partial design, and the Design Exploration Analyzer allows software designers to explore a set of users’ partial designs.

4. DESIGN EXPLORATION BUILDER

Many tools, including pencil and paper, could be used to collect partial designs. However, collecting this information in an electronic format will enable computational support for software developers as they explore the set of partial designs communicated by a set of potential users.

Existing GUI construction kits available in many programming environments and prototyping tools provide one avenue for collecting end user expression. However, it is important that respondents' partial designs not get bogged down in the superficial details that make design of the actual artifact time consuming. The time respondents are able and willing to provide is limited, even if they are compelled to respond by an external force. The building tool should keep the expressive actions of the respondents at a level that will be useful to software designers, such as identifying features and alternatives they believe are necessary in the interface, rather than other activities, such as making sure widgets are precisely aligned and distributed. A tool for expressing partial designs should result in partial designs that are rough-hewn graphical user interfaces since the tool intentionally limits the ability to fine-tune the interface, e.g. change fonts and colors. This low fidelity constraint helps users focus on the information and not become engrossed with aesthetics [Landay and Myers 1995].

An existing tool that could be adapted to collect user partial designs and epitomizes the low fidelity constraint is SILK [Landay and Myers 1995]. SILK allows designers to sketch interfaces using an electronic pad and stylus. However, most users do not have access to the physical set up required for sketching since it is difficult to sketch

using a mouse. Establishing a location where users could go to access this set up would re-introduce issues related to scheduling user time that this approach attempts to circumvent.

In Design Exploration, users are trying to communicate program behavior. Programming by demonstration (PBD) systems do something similar. They move beyond macros and not only allow users to express how software is to behave in a single context, but also discerns a generalized program [Cypher 1993]. PBD is characterized by the domain, how users interact to create programs, how the system infers the generalize program, and the information used to do the generalization [Cypher et al. 1993]. However, these systems are created for specific domains that limit the range of possible actions. Moreover, users must learn how to interact with the PBD system to "program" it. Since the Design Exploration process could be used to gather information from a variety of domains and the collection of partial designs should require a minimal of learning, PBD is not a viable approach for Design Exploration.

4.1 User Interface

The Design Exploration Builder was developed as a Java application. Users begin their design process with a blank slate to avoid seeding them with ideas (Figure 2). Pressing the Create Window button creates a blank window (Figure 3) and adds a reference to it in the overview window (Figure 4). These created windows can be closed at anytime and reopened by double clicking on the window listed in the overview.

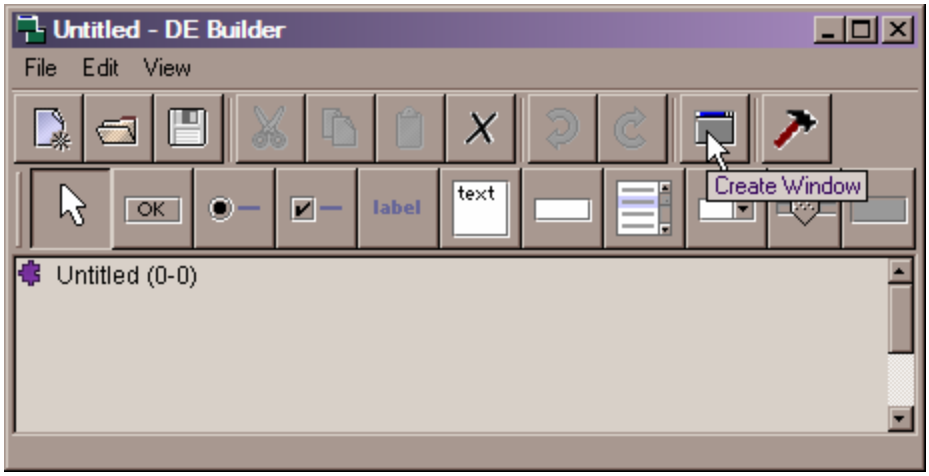


Figure 2. Design Exploration Builder.

Figure 3. Newly created window.

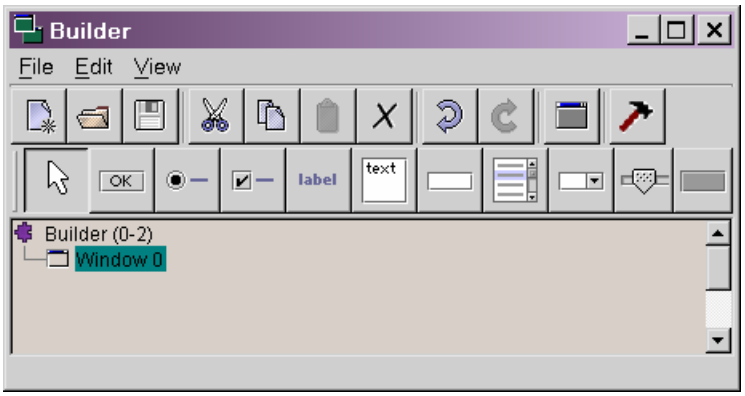


Figure 4. Design Exploration Builder overview containing reference to a window.

The widget pallet allows users to add a variety of widgets (Table I). Note the generic widget that can represent functions, such as “image”, that are not in the available widget set. This combination supports the expression of interface designs that are easy to develop using standard widgets and those that are not. The widget pallet toggles between each widget and the selection arrow. For example, selecting the radio button pallet item and clicking in the blank window adds a radio button to that window (Figure 5). Double clicking on a widget or on the background of a window will open an editor for the widget or window respectively. One tab is for editing attributes of that widget/window (Figure 6), and the other tab is for providing argumentation (Figure 7). Alternatively, when adding a widget, a rectangle can be dragged that defines a region that the widget will fill when created (Figure 8).

Table I. Design Exploration Builder widgets

	Push Button
	Radio Button
	Check Box
	Label
	Text Area
	Text Field
	List
	Combo Box
	Slider
	Generic Widget -- a grey rectangle used as a place holder for something the user wants to include that is not one of the widgets available --

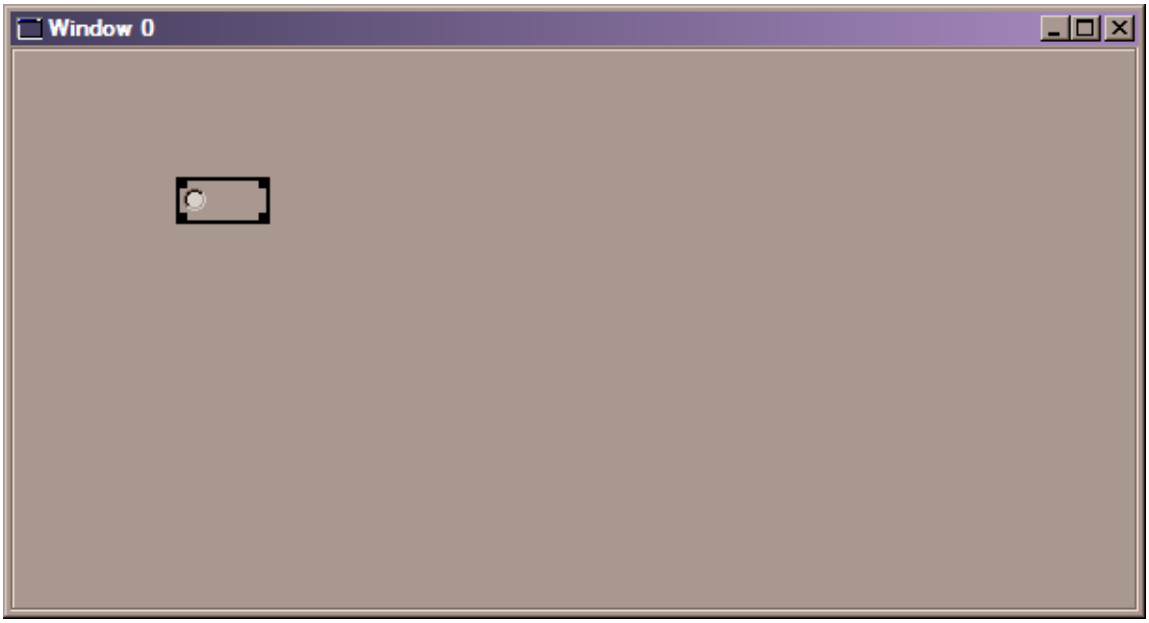
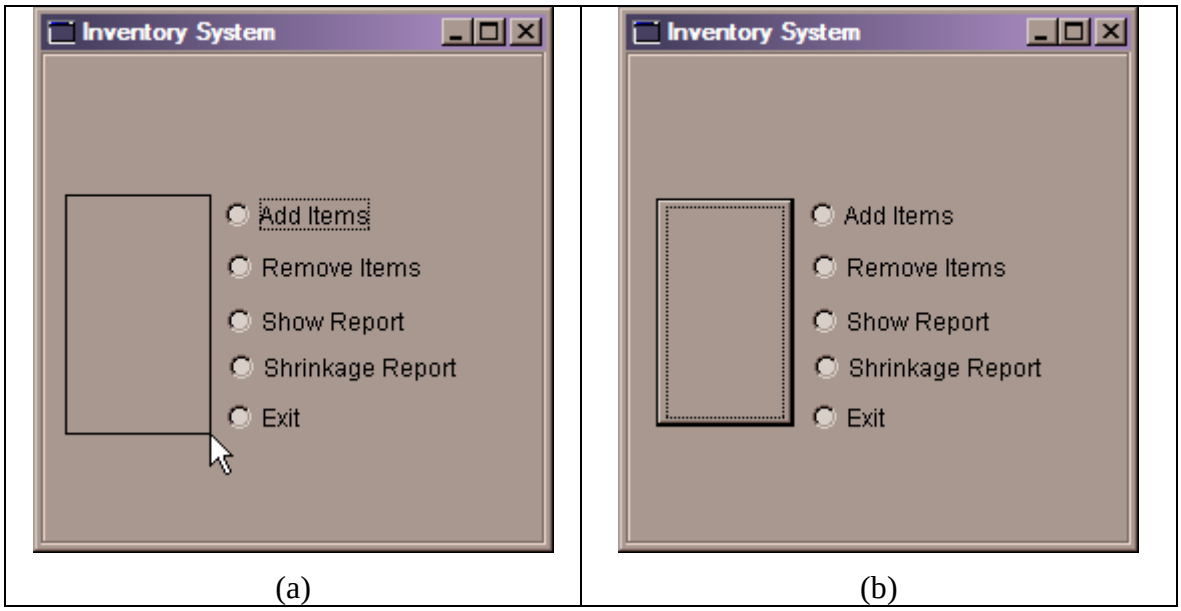


Figure 5. Window with radio button added.

Figure 6. Label for radio button added.

Figure 7. Argumentation for window.



(a) (b)

Figure 8. Adding a widget by dragging.
(a) Dragging region for widget (b) Widget added with drag

Widgets can be resized and moved. When a widget is selected, selection marks are shown. These can be clicked and dragged to effect a resize (Figure 9). Widgets can

be moved individually by selecting the widget and dragging it from inside the selection marks (Figure 10). A group of widgets can also be selected and moved as a group (Figure 11).

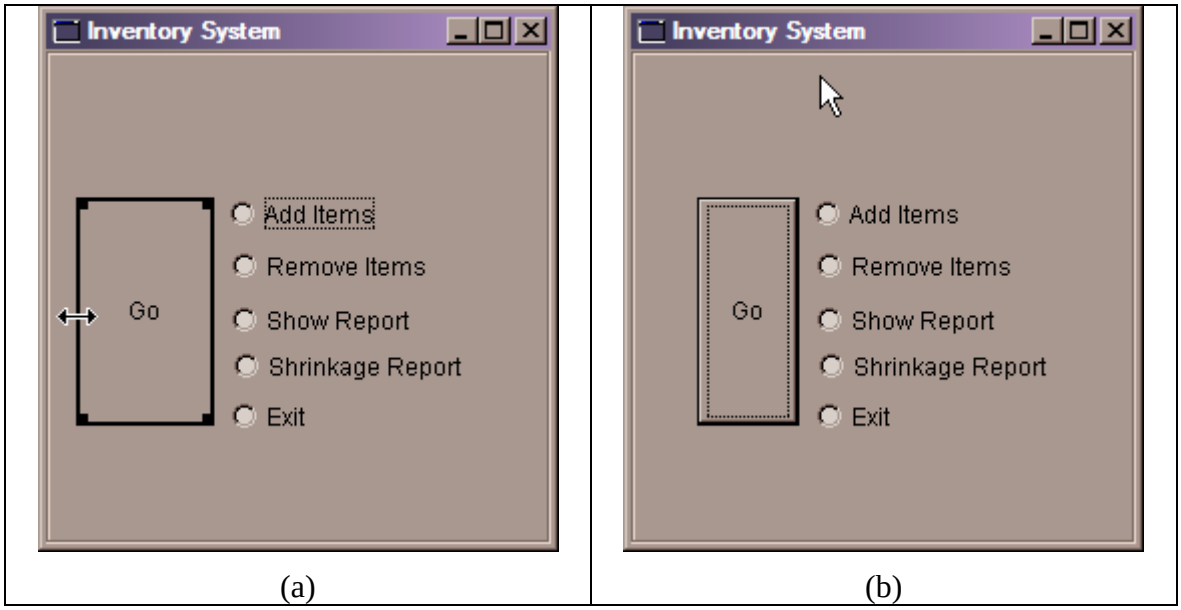


Figure 9. Resizing a widget.
(a) Widget with left edge grabbed for resize (b) Widget after resize

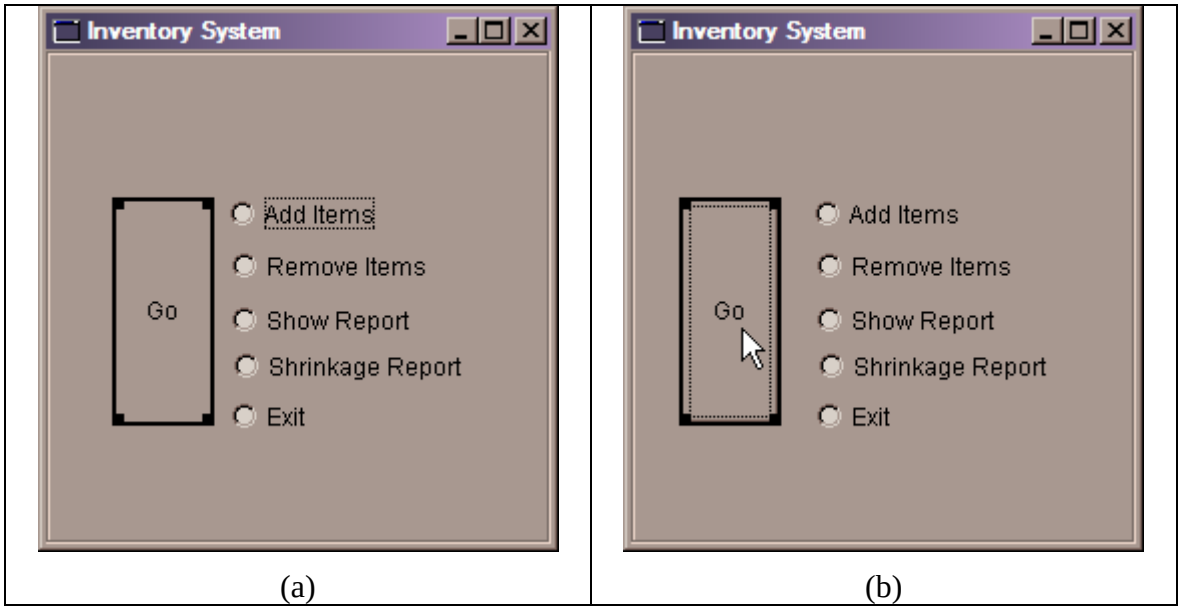


Figure 10. Moving a widget.
(a) Widget selected for move (b) Widget at end of move

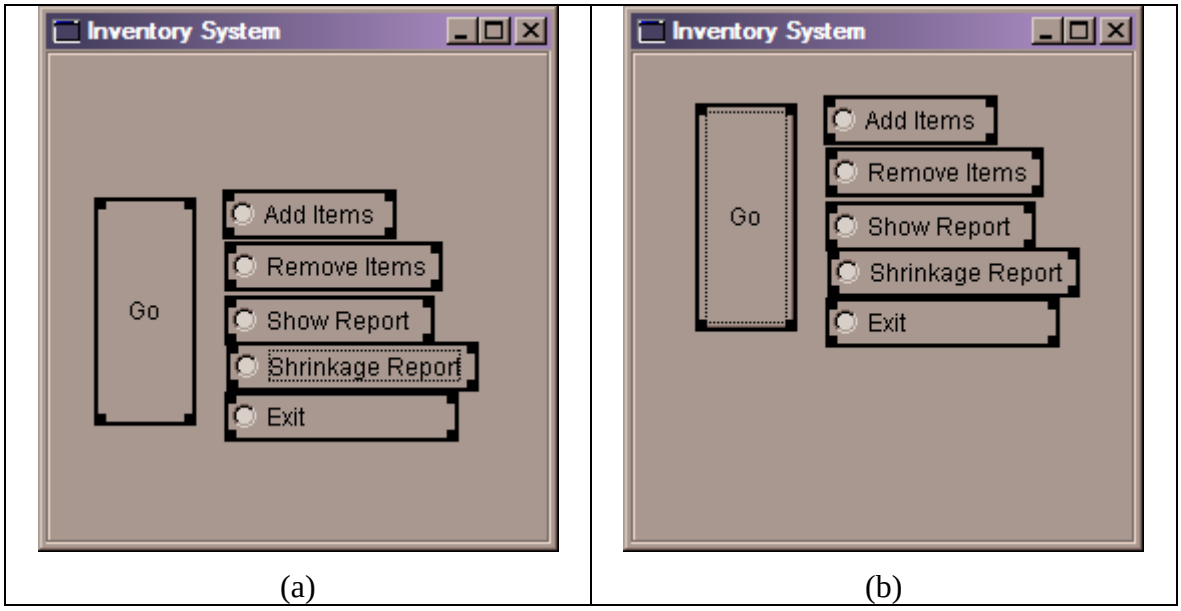


Figure 11. Selecting and moving a group of widgets.
(a) Group of widgets selected (b) Group of widgets after move

4.2 History

As partial designs are created, a comprehensive history is maintained as is done in systems such as INDY [Reeves 1993] and VKB [Shipman and Hsieh 2000]. However, in a creation process users may wish to backtrack to an earlier point to generate an alternative design. Linear history systems lose the history in the abandoned branch. Branching history solves this problem by creating a new branch in history when backtracking and starting work in a new thread. While abandoned branches may not embody the final results the user wants to portray, they might provide pointers to information the user thought was important at one point but decided to abandon in their alternative design. Moreover, an abandoned branch could represent the attempt to express a desire that the user could not figure out how to express thereby providing a hook into further exploration for possibly tacit knowledge. By preserving all aspects of history, abandoned paths can be explored in much the same way as alternative scenarios are viewed in Visage [Derthick and Roth 2001].

4.3 Design Exploration Builder Summary

The Design Exploration Builder provides a simple interface for creating rough mock ups to express users' desires for software. Providing a simple non-commercial tool allows a large number of users to be involved in the Design Exploration process while not requiring the time and co-location of face-to-face meetings. Moreover, avoiding the ability to polish designs, as would be the case with many interface design environments, encourages users to focus the basic information rather than aesthetics.

4.4 Summary

The Design Exploration Builder provides a simple interface for creating low fidelity design mock ups with annotations. The tool has very few features to constrain expression to content rather than aesthetic characteristics.

5. DESIGN EXPLORATION ANALYZER

Users create a large volume of communication through the Design Exploration paradigm. Consequently, software designers need the ability to explore partial designs without having to view all aspects of each design. The Design Exploration Analyzer uses textual analysis and spatial parsing of the artifacts created by users to provide software designers browsing, search overlay, and navigation options.

Communication through design creates information that is expressed both textually and spatially through the layout of windows and widgets. To do textual analysis, germane semantic units of text must be identified.

Groupings of design components for textual analysis are determined in three ways. First, each widget and window has text associated with it through annotations and any widget data text such as the title for a window, the label on a button and items in a list. Second, windows represent data for the window itself as well as for all of widgets that it contains. So windows and their included widgets can be combined as a textual group. Thirdly, since windows can be complex entities, sub-groups of widgets within a window can be combined to form a textual group, e.g. a list of radio buttons. Spatial parsers can help identify these sub-groups.

5.1 Term Vectors

The vector space model for terms, or term vectors, provides a relatively simple way to assess textual similarity between documents. Similarity is assessed by taking the cosine between the term vectors representing each document [Witten 1999].

Designers will often want to find things that are similar. In most cases this similarity is textually based. For a given widget similar items could include other widgets, windows including the widgets contained in them, and groups identified by a spatial parser. Accordingly, term vectors are created representing these potentially similar items (i.e. documents).

For comparison, these term vectors are analyzed as a set. This allows for term normalization across the set of term vectors. Cosine similarity relies on the total frequency of a term, the number of documents that contain that term, and the total number of documents. The total frequency for a term is determined using the term dictionary (described below). The number of documents that contain a term is based on how many design components contain the term, and the total number of documents is determined by the number of design components that have some attached text.

Given a particular term vector, e.g. from a grouping found while exploring the space, a list of similar items can be extracted from the term vector set. This is the basis for the right click navigation described in the scenario presented in section 5.5 (Figure 30).

5.2 Spatial Parser

Each window in a design can have multiple purposes. Figure 12 shows a window containing multiple distinct areas created by a user in the study described in Section 6. Each area represents different aspects of the domain. Sub-groups from different windows might be similar; however, considering only windows for comparison could

miss finer grained connections since similarity between the two windows as a whole would be diluted from terms in other areas of the windows.

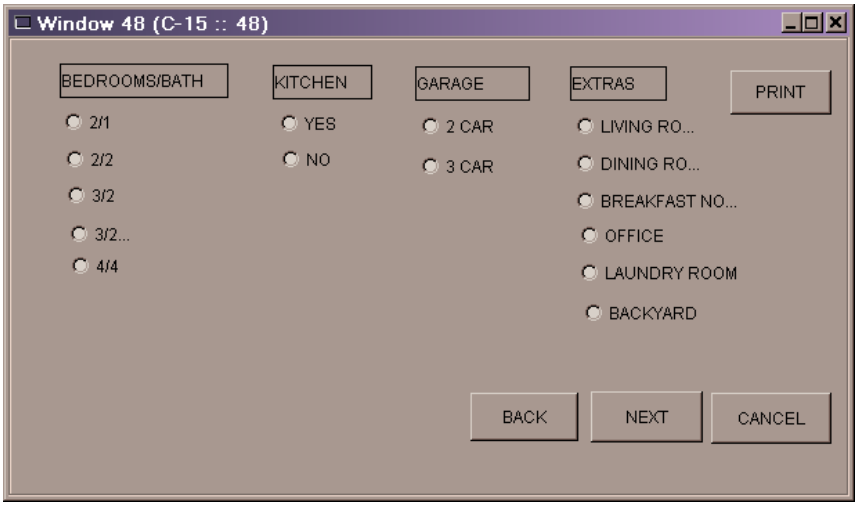


Figure 12. Partial design with distinct areas.

Spatial parsers can identify related groups of objects based on their spatial arrangement [Shipman et al. 1995]. The Design Exploration Analyzer uses the spatial parser from VIKI and VKB [Shipman et al. 2001b; Shipman et al. 1995] to segment windows created by users to facilitate analysis. Spatial parsers look for patterns in the spatial layout of objects to identify orderings of objects such as lists and stacks. Figure 13 shows a user design with a tree view of a hierarchical spatial parse. The lists of radio buttons are identified as vertical lists. While the parser is capable of identifying the pattern of a text area over a vertical list of radio buttons, the parser is currently loaded in a way that treats all widgets as equivalent types. The parser also identifies that there are two vertical lists comprising a composite. Parsers are not perfect. And spatial parse

structures can be quite complex. Determining how to break the parsed tree into meaningful units is a challenge. In this case combining composites actually lessens the effectiveness of parsing. The items in each list of radio buttons are related; however, the different lists are only related in the broader context of the inquiry form that the window represents. Currently, the Design Exploration Analyzer limits its analysis to spatial groupings that contain only leaves, i.e. only widgets and not groupings that include composites. Figure 13 is part of a window created by a user in the study presented in Section 6. Two spatial groups of radio buttons and the text field above them are identified as design components for analysis.

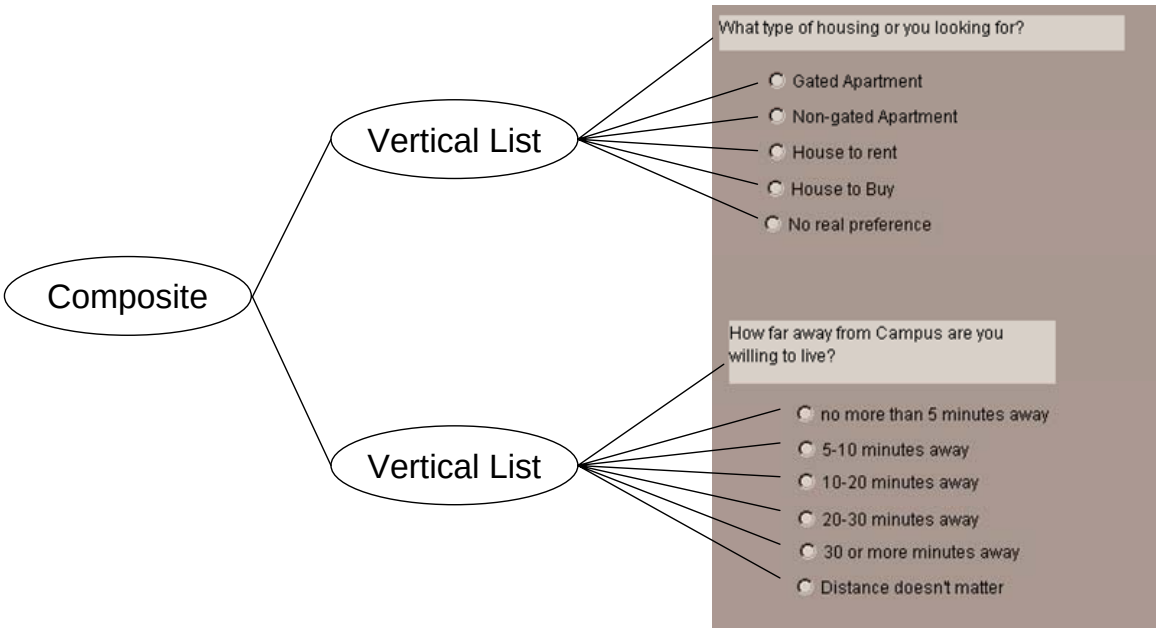


Figure 13. Part of user designed window with tree representation of spatial parse.

5.3 Clustering

Clusters provide a starting point for examining similarity between user partial designs. All clustering is based on textual similarity. Three sets of clusters are provided. In the first, individual widgets and windows are clustered. In the second, windows including text from their widgets are treated as a unit for clustering. Finally, the last clusters based on a loose container concept called design components. Here a container can contain multiple items. Naturally windows fall into this category. Secondly, groupings identified by the spatial parser are included (note: compound groupings are not used). Finally, list and combo box widgets are included since they contain sub-items.

The Design Exploration Analyzer clusters using hierarchical agglomerative clustering where term vector cosine similarity is the distance metric. For each unit used in clustering, a term vector representation is created by combining the term vectors for all of the relevant widgets and windows.

5.4 User Interface

The main interface for the Design Exploration Analyzer has four major areas: a tree view on the left, a tree control under the tree view, an information panel on the right, and a search overlay beneath the information panel (Figure 14). The tree view provides a hierarchical view of the content and can be switched between views of partial designs, terms, clusters, and spatial groups.

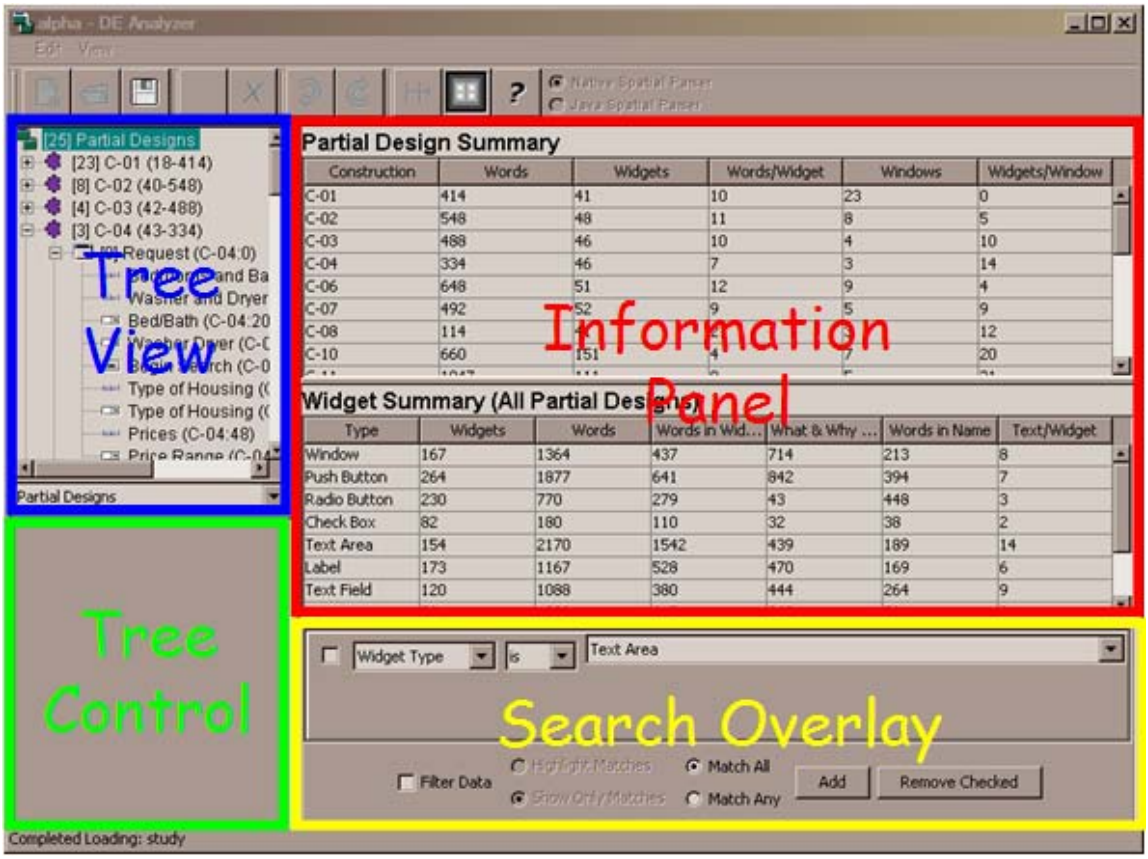


Figure 14. Main interface with four areas.

The icons and labels for the four tree views are shown in Table II. The tree control provides ways of modifying the ordering of items in the tree. Options are only available in the terms and clusters views. The information panel shows information for the specific item selected in the tree view. Finally, search overlay allows for viewing search results in the context of the tree view rather than a separate list.

Table II. Tree view icons and labels for four views

View	Item Type	Icon	Tree Node Label
Partial Designs	Exploration Set		[**X**] Partial Designs • **X** - number of partial designs in the set.
	Partial Design		[**X**] **Name** (**Y**-**Z**) • **X** - number of windows in a partial design • **Name** - name for the partial design • **Y** - number of widgets and windows in the partial design • **Z** - number of terms found throughout the partial design
	Window		[**X**] **Name** (**ID**) • **X** - number of widgets in the window • **Name** - name for the window • **ID** - unique identifier for the window
	Widget	See Table I	**Name** (**ID**) • **Name** - the name for the widget • **ID** - unique identifier for the widget
Terms	Term		[**X**] **Term** (**Y**) • **X** - number of widgets and windows containing the term • **Term** – term • **Y** - overall frequency of the term
	Window		**X** in **Name** (**ID**) • **X** - number of term occurrences in the window or widget • **Name** - name for the window or widget • **ID** - unique identifier for the window or widget
	Widgets	See Table I	
Spatial Groups	Vertical List		[**X**] **Type** in **Window** (**WindowID**) • **X** - number of widgets in the spatial group • **Type** - type of spatial group • **Window** - name of the containing window • **WindowID** - ID for the containing window
	Horizontal List		
	Composite		
	Group		
	Others function in the same way as in Partial Designs.		
Clusters	Cluster		[**X**] Cluster **Y** • **X** - number of items in the cluster • **Y** - cluster ID
	Others function in the same way as in Partial Designs.		

5.4.1 Partial Designs

The information panel shows all of the information collected by the Design Exploration Builder. Figure 15 shows the information panel for a single widget. The name of the widget provided by the user is shown at the top (i.e. “Apartment Button”). Underneath the name is a thumbnail view of the window containing the widget and text associated with the widget. When representing a widget, the widget in the thumbnail is outlined in red while.

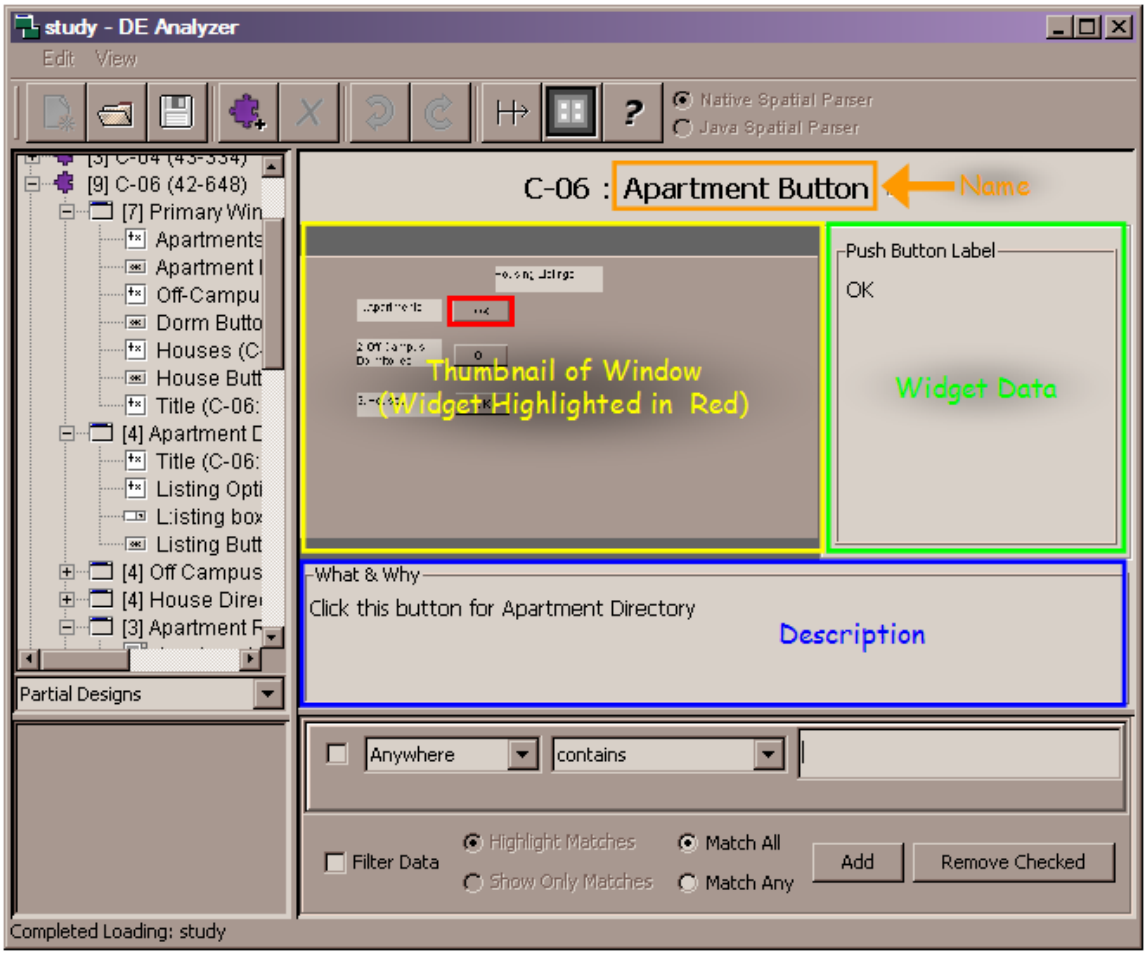


Figure 15. Information panel for widgets and windows.

The information panel for a partial design shows the titles and thumbnails for each window contained in the partial design. Figure 16 shows a partial design made up of five windows.

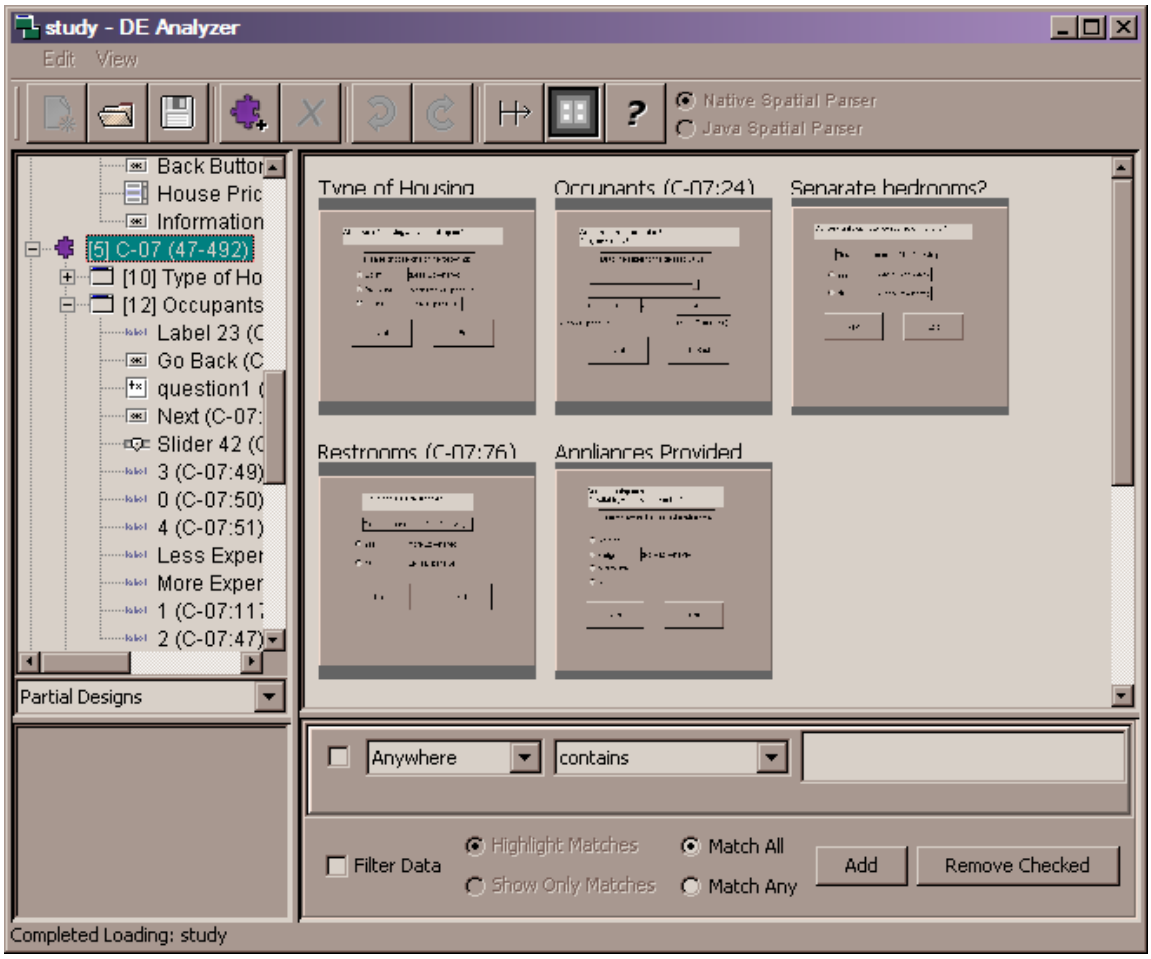


Figure 16. Information panel for a partial design.

The information panel for an exploration set, i.e. the grouping of all partial designs being explored, shows information about each partial design as well as some aggregate information (Figure 17). This table provides information on the number of

widgets and windows produced and the amount of text generated in annotations and widget data.

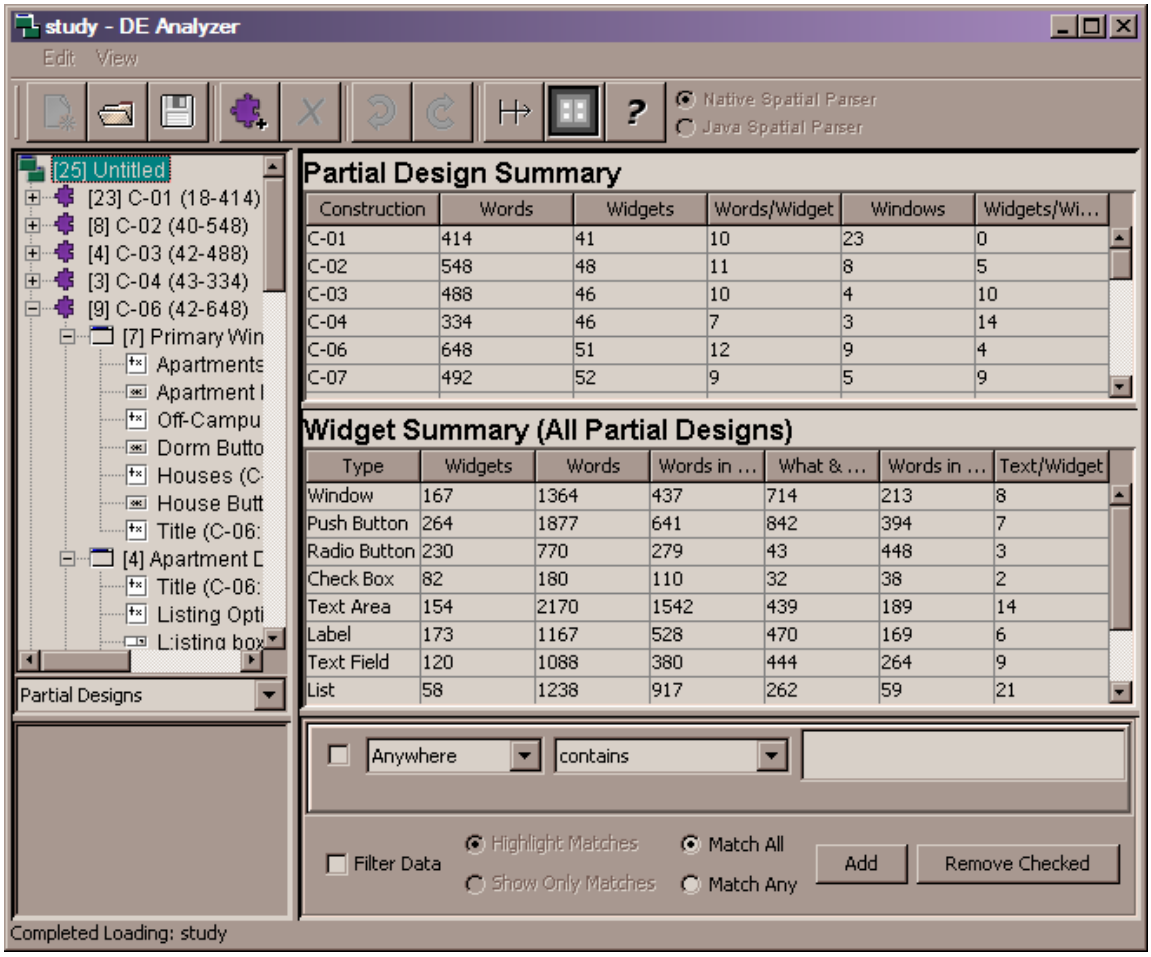


Figure 17. Information panel for exploration set.

5.4.2 Terms

The Design Exploration Analyzer maintains a dictionary of each term used in a partial design along with references to the windows and widgets. This allows for easy

browsing to partial design elements where a term of interest can be found. The tree view shows the term with its associated windows and widgets (Figure 18).

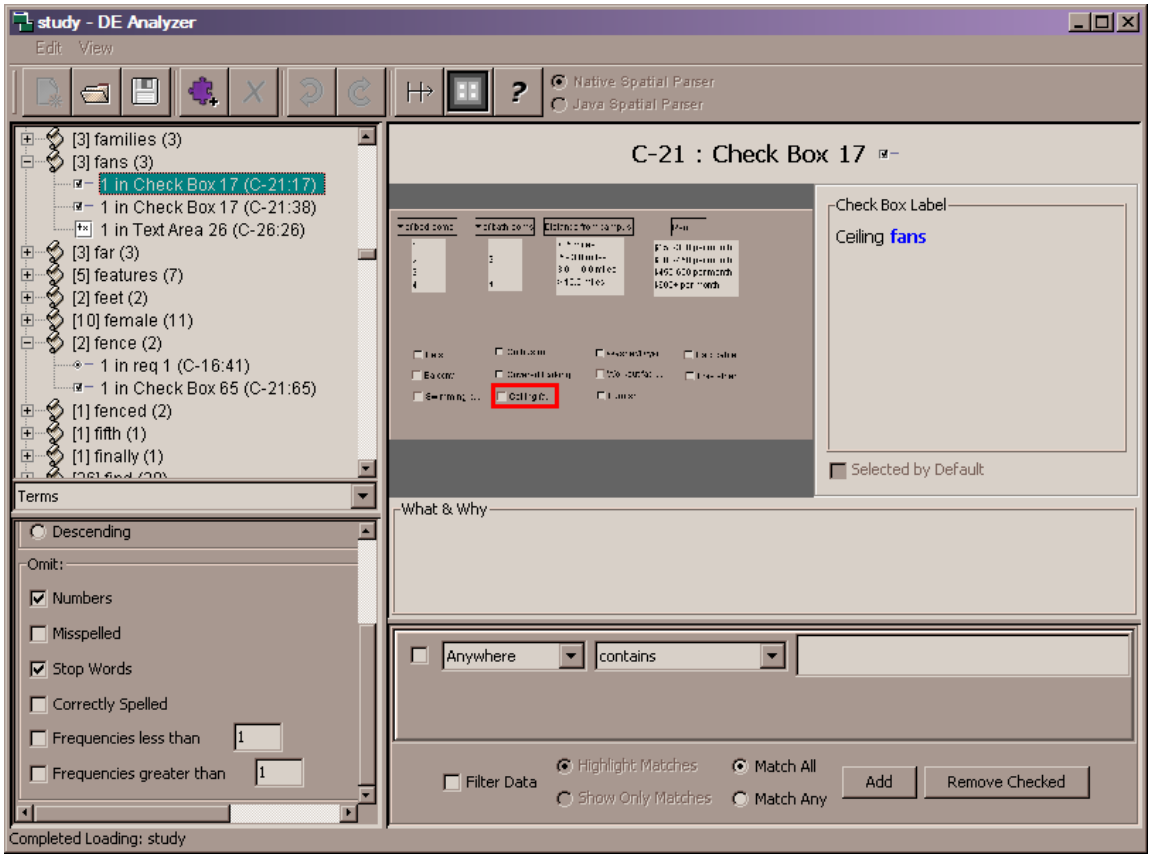


Figure 18. Information panel for term in a window or widget.

The tree control allows the display of the terms to be altered (Figure 19). Terms may be ordered alphabetically or according to frequency. When ordered by frequency, terms with a low frequency can indicate unique concepts that designers may want to further explore. Terms with a high frequency across a number of partial designs indicate common domain concepts. Stop words, terms that are part of the structure of a language

and are not related to the semantic content e.g., “a”, “an” and “the”, will also appear frequently. Within the Design Exploration process, additional stop words exist for the language of the GUI (Appendix A). Examples of these are “button” and “click.” These terms appear more frequently because they describe user interface elements and actions, but do not deal directly with domain concepts. The tree control allows the software designer to pare down the list of terms shown to reduce clutter in the list. Terms can be omitted from view if they are numbers, if they are stop words, if they are correctly or incorrectly spelled (via an integrated spell checker), and if they occur above and below user defined frequencies.

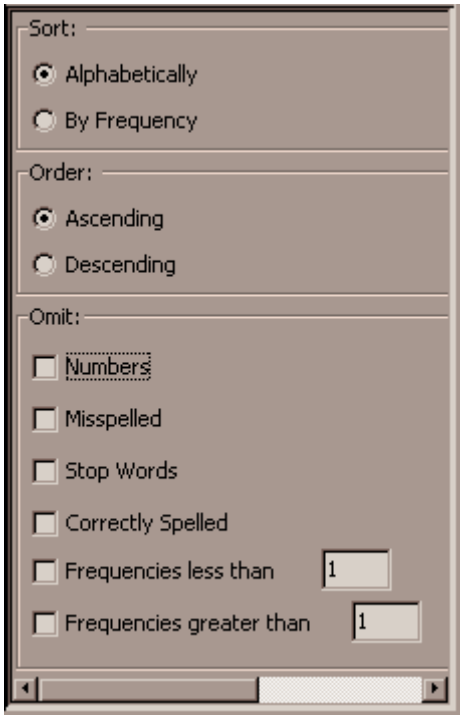


Figure 19. Tree control for terms view.

The information panel for a term provides details about the distribution of the term (Figure 20). Clicking on the "Definition" button will open a web browser with a Google definition search for the term.

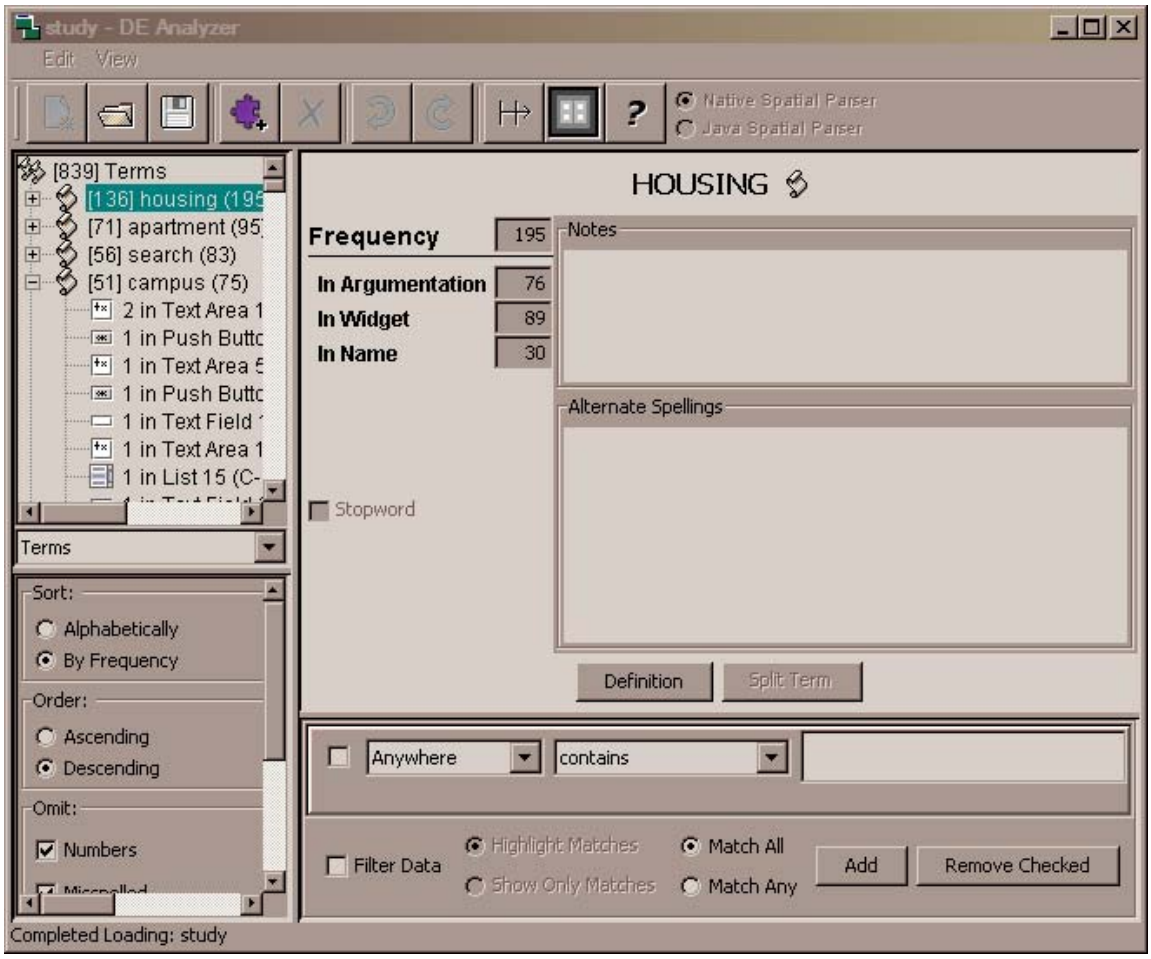


Figure 20. Information panel for terms.

The information panel for widgets and windows in the terms view is identical to the one in the partial design views except that the term is highlighted with a blue font where it appears in the panel (Figure 21).

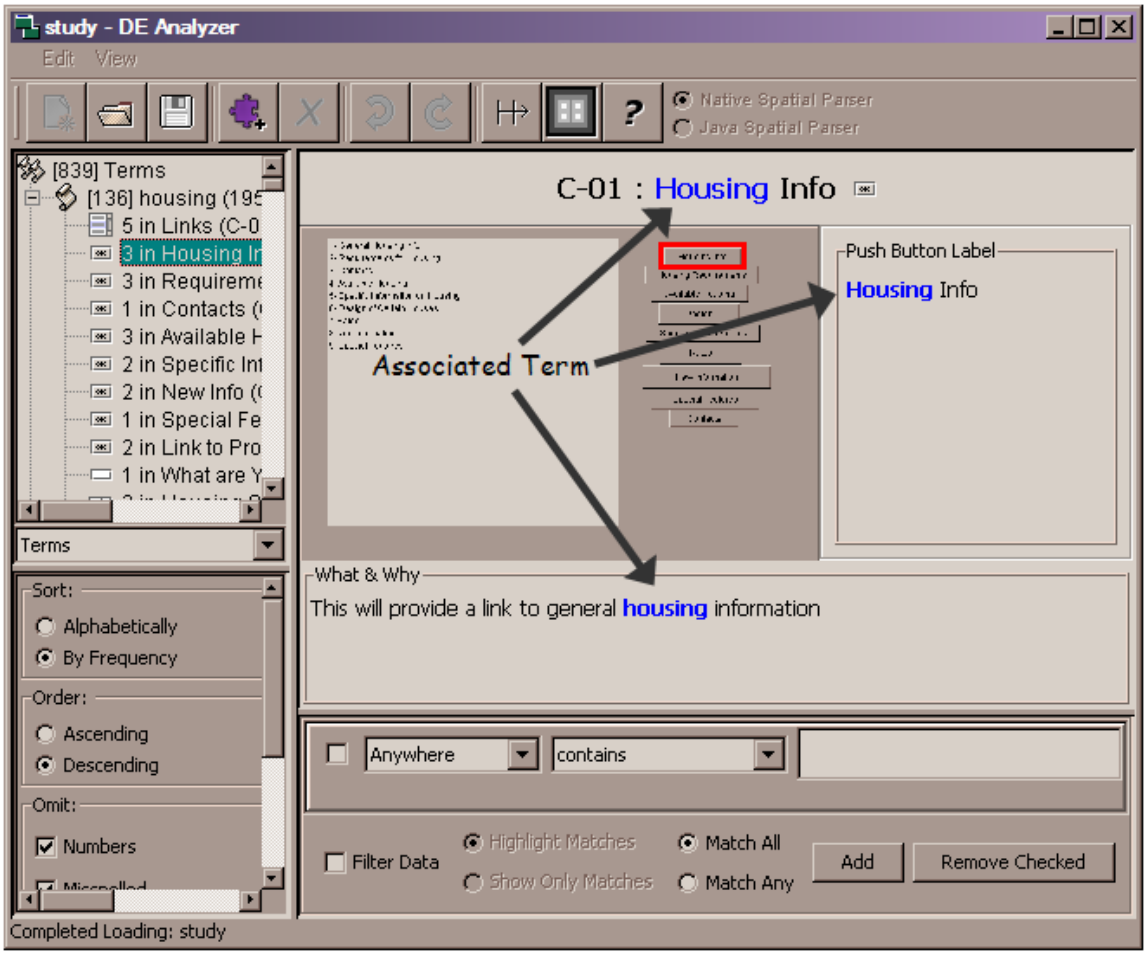


Figure 21. Information panel for widgets and windows in terms view.

5.4.3 Spatial Groups

The information window for spatial groups shows the widgets constituting the grouping in the context of their window (Figure 22). The items that are part of the selected group are highlighted in green in the thumbnail. All other information panels for this view function as they do in the partial designs view.

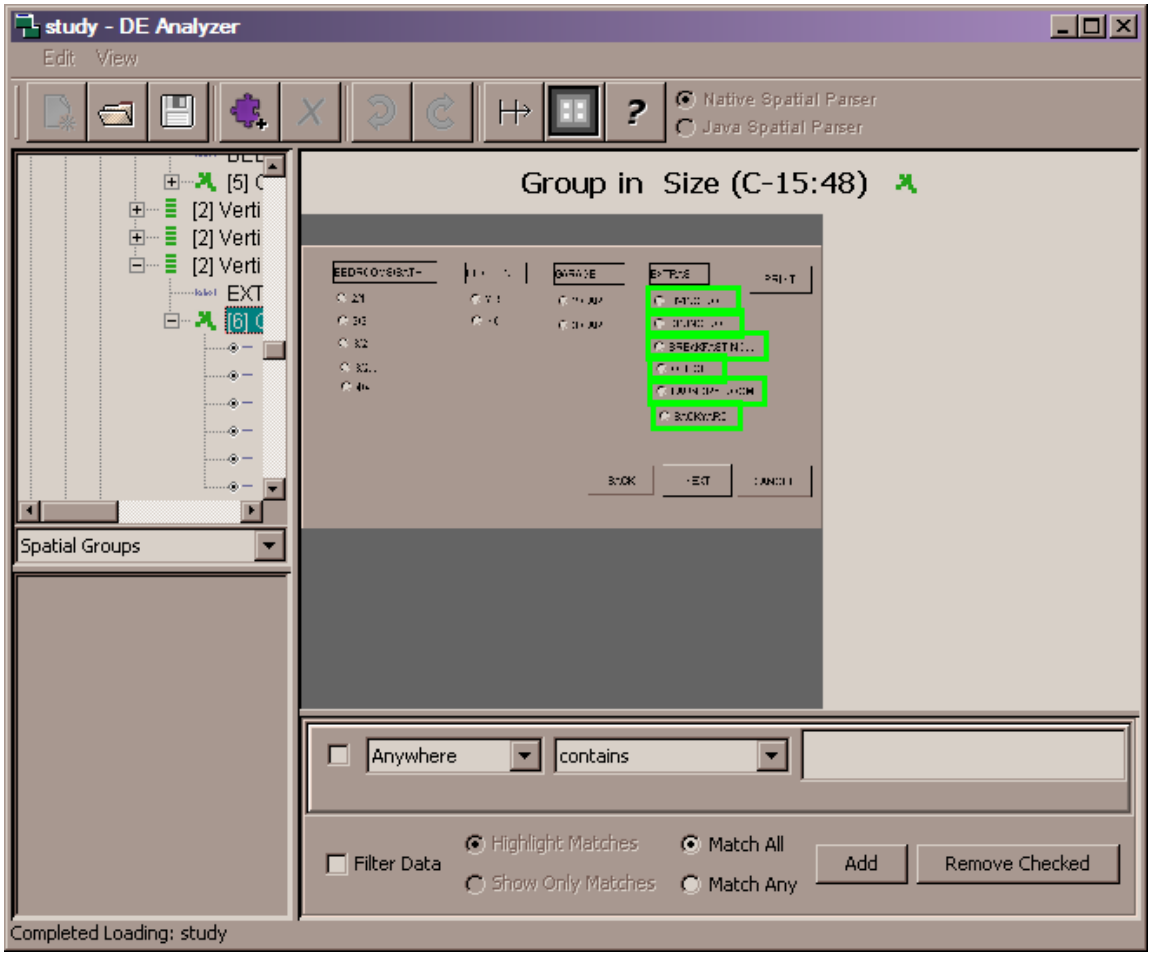


Figure 22. Information panel for a spatial group.

5.4.4 Clusters

The information window for cluster groups shows thumbnails for each item in the cluster (Figure 23). These representations all occur in the context of a window. When representing a widget, the widget is outlined in red. When representing a spatial group, all of the widgets in the group are outlined in green. When representing a window no widgets are outlined. After items of interest are identified, the more fine grained

navigation provided through right clicking can be used to find other similar design components and groupings.

All other information panels in this view function as they do in the partial designs view and the spatial groups view.

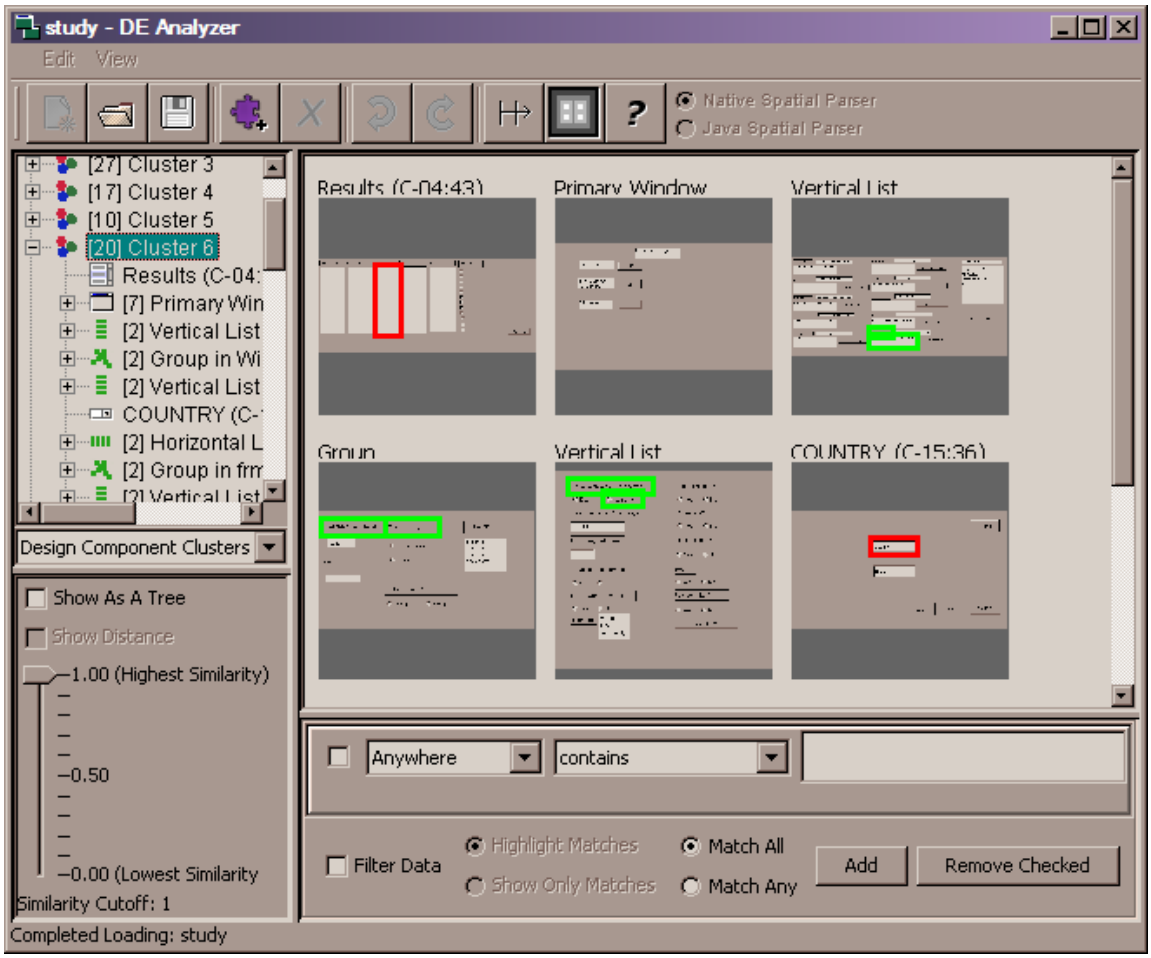


Figure 23. Information panel for a cluster.

The tree control allows changes in the way the clusters are viewed. Since hierarchic agglomerative clustering begins by creating a tree that is then sliced at various

distances (i.e. similarity), the clusters can be viewed as trees and the distance cutoffs can be changed via a slider.

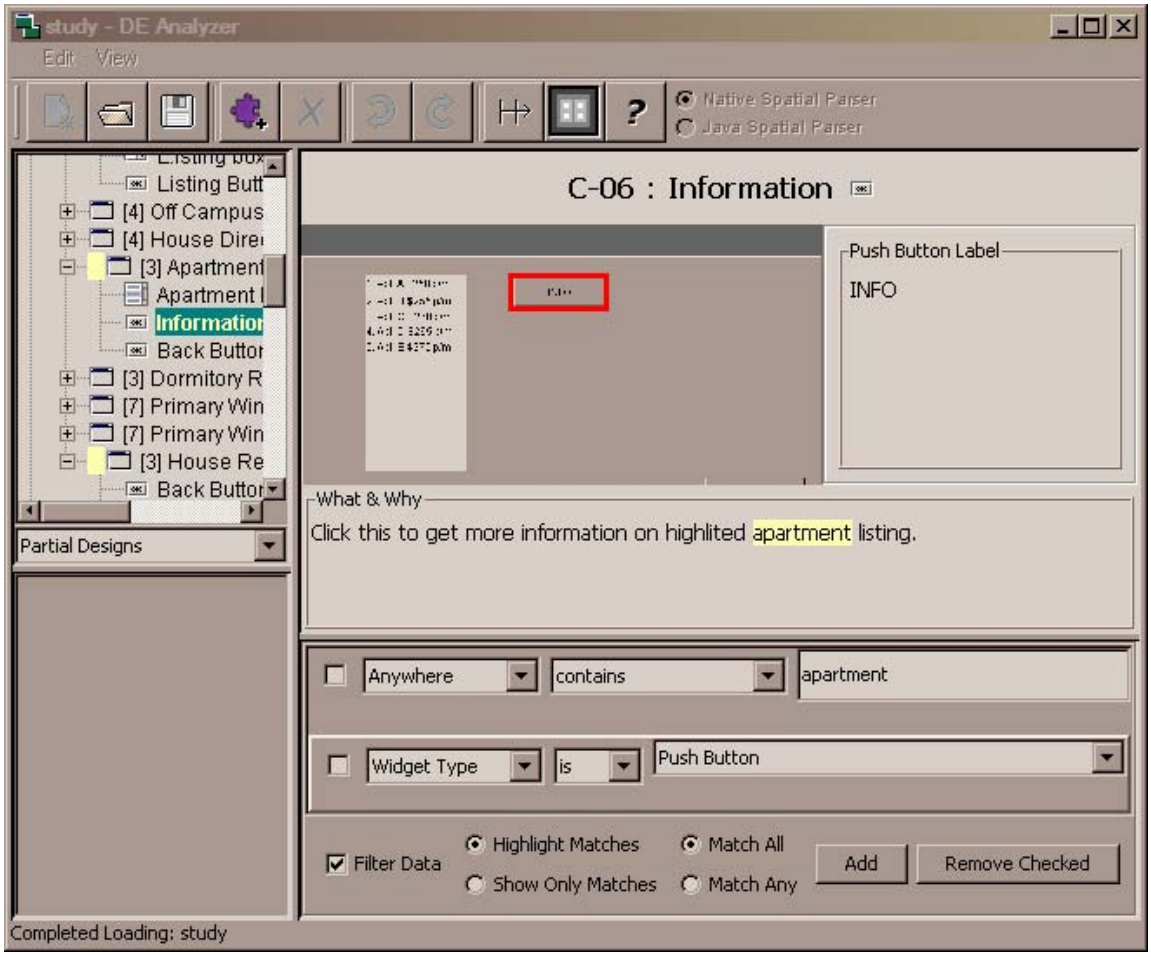


Figure 24. Search overlay with results highlighted.

5.4.5 Search Overlay

The search interface supports queries based on textual content, types of widgets, and for objects in groups identified by the spatial parser. As designers work to understand the domain and users' perspectives on interaction, the search overlay can

help identify interesting intersections of various characteristics that can be expressed as a criterion for the search.

Rather than the traditional approach of providing a list of matches, search results are overlaid onto the larger context of the view by highlighting results in the tree view and information panels (Figure 24). Alternatively, the results can be limited to showing only items matching the search criteria, thereby hiding all non-matching items (Figure 25).

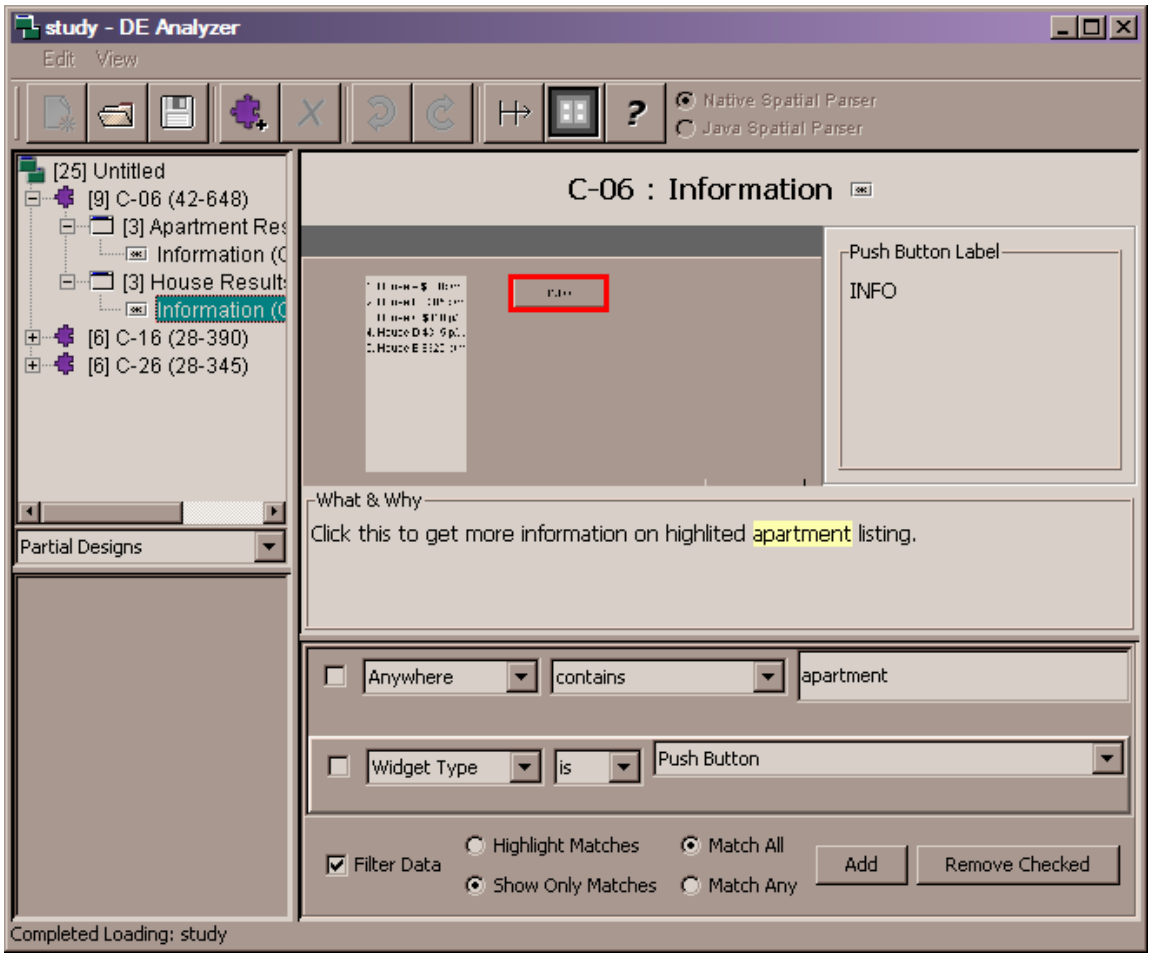


Figure 25. Search overlay showing only matches.

Since results are displayed in a tree view where windows can not only be a match but contain a match as well. Consequently, nodes must be marked in a way to distinguish between these two cases. Items with text on a yellow background match, and items with a yellow square to the left of the icon contain a match (Figure 26).

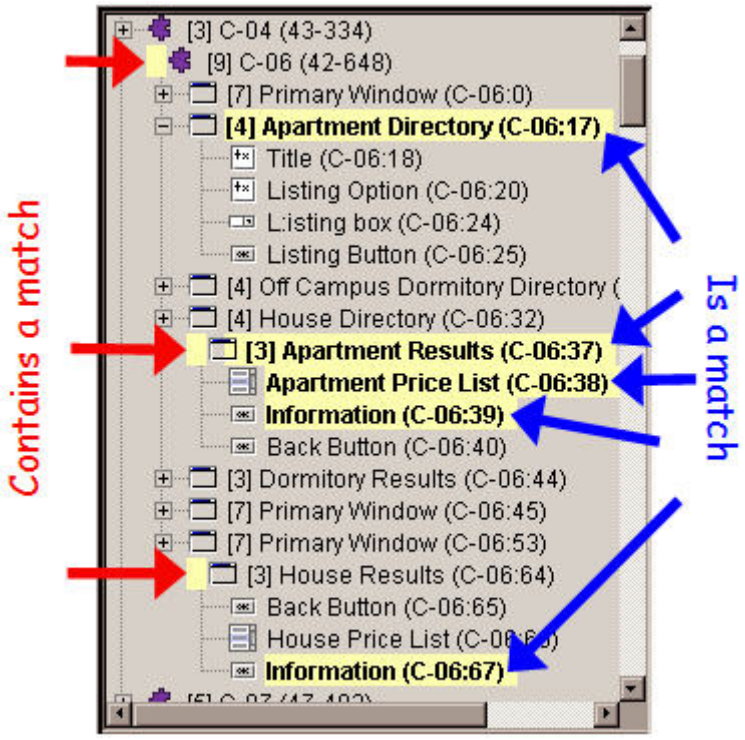


Figure 26. Search overlay effects on tree view.

5.4.6 User Windows

Thumbnails are not sufficient for viewing partial designs. User designed windows can be opened in several ways. Double clicking on a widget, window, or spatial group in the tree view or a thumbnail will toggle (show/hide) the associated window. Right clicking an item in the tree view or a thumbnail in the information panel

will pop up a menu with options to show and hide windows. Doing this on the partial design node offers a quick way of opening/closing all of the windows in a single partial design. At the same time doing this for the exploration set opens all windows which can be overwhelming.

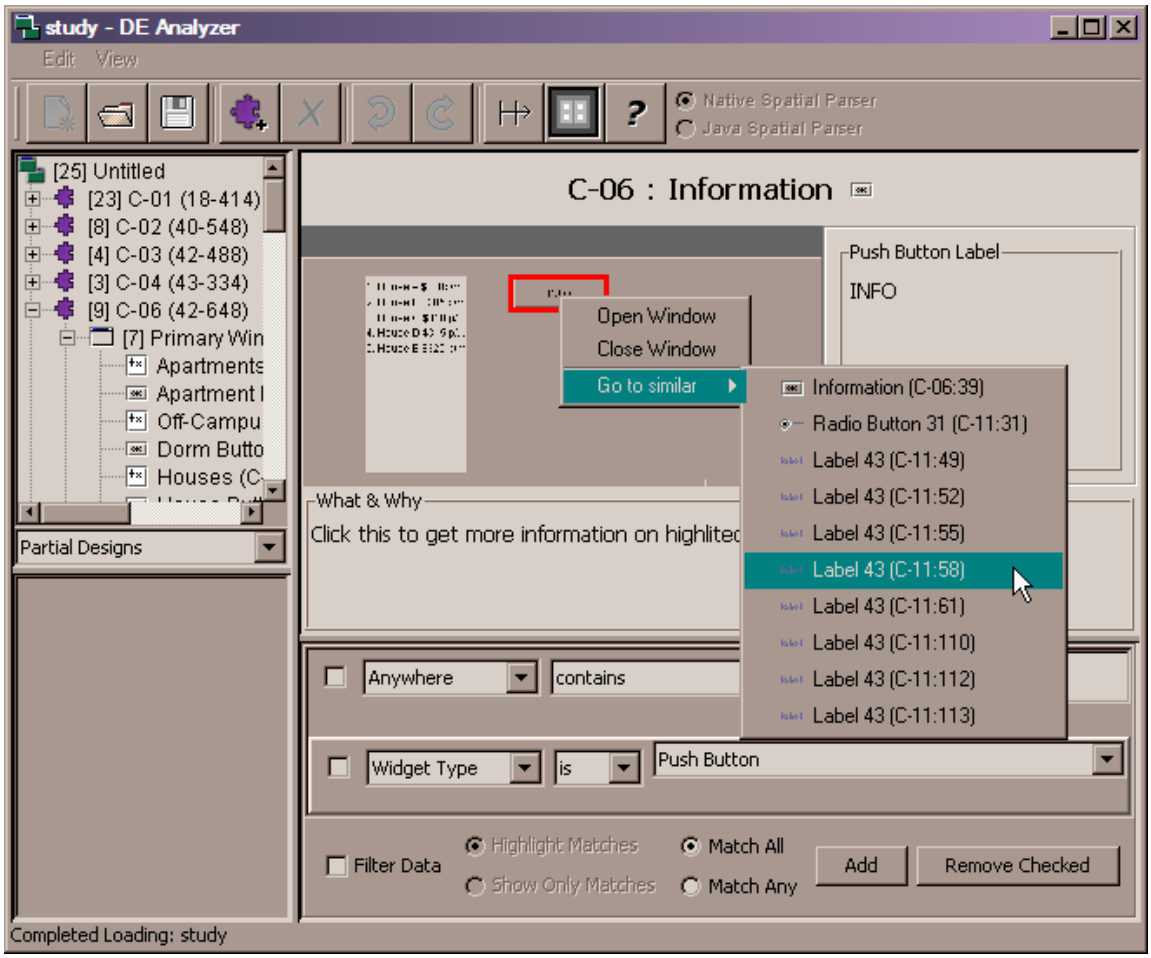


Figure 27. Similarity navigation.

5.4.7 Similarity Navigation

All windows (including its widgets), individual widgets, and spatial groups are textually compared to provide similarity navigation. When right clicking on a thumbnail or an item in the tree view, the ten most similar items are provided as navigational options (Figure 27).

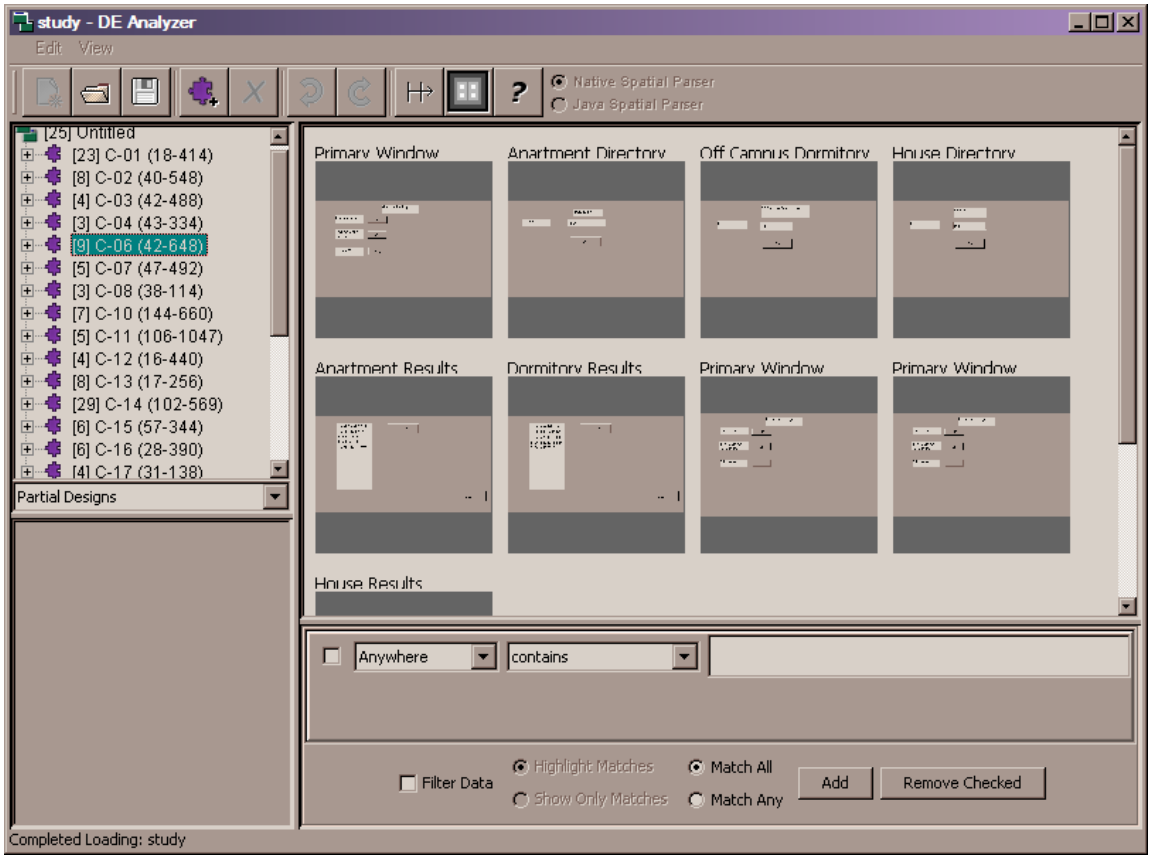


Figure 28. Thumbnails shown for user C-06 partial design.

5.5 Scenario

The following scenario will describes an example of the process of using the Design Exploration Analyzer to explore a set of annotated partial designs.

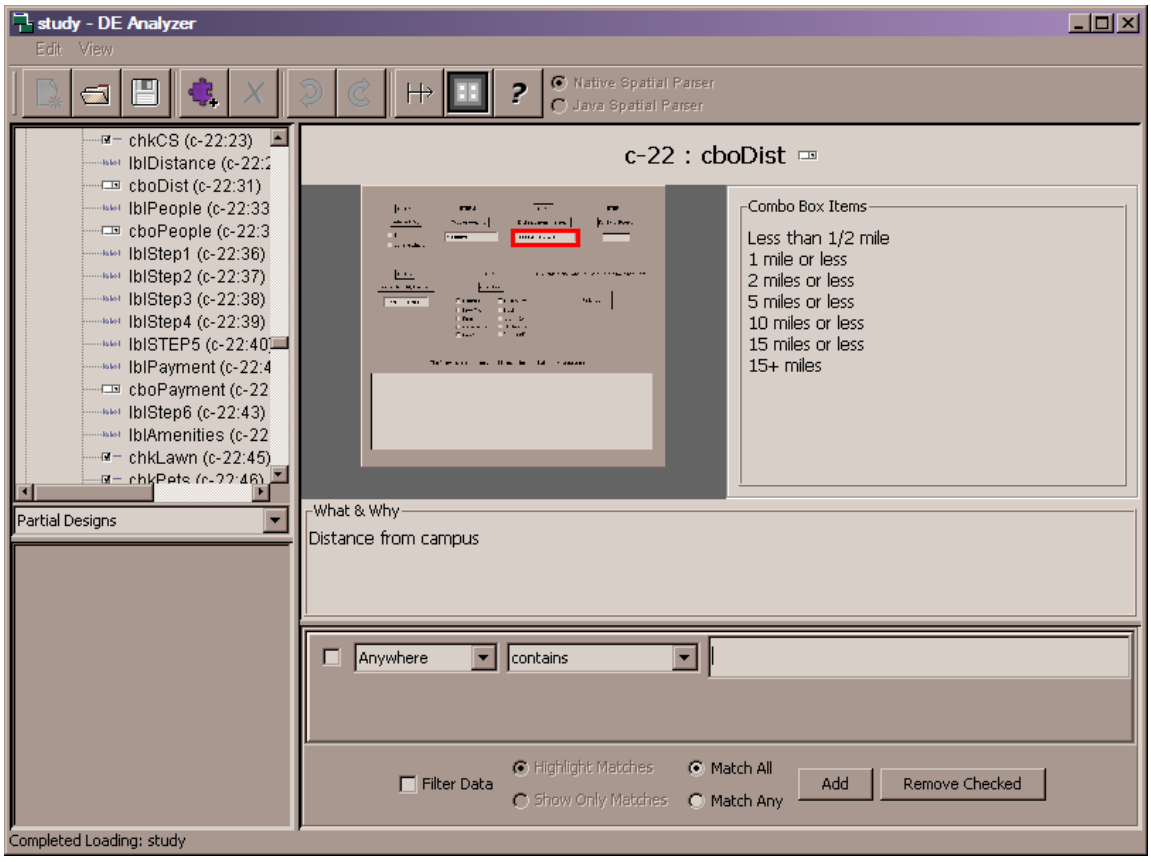


Figure 29. Distance information in a single combo box.

A software designer is starting work on the company's project to develop software to help college students find housing. After collecting responses created with the Design Exploration Builder from college students, the software designer loads the set of partial designs created by the students (identified as C-1 to C-30). Clicking on the

puzzle icon, which represents a student's partial design, displays thumbnails of all of the windows in the partial design. Figure 28 shows thumbnails for windows in the C-06 partial design.

To get an initial overview of various students' partial designs, the software designer looks at groups of windows clustered based on textual similarity (Figure 23).

During this exploration, the designer finds a combo-box in one partial design showing distances from campus (Figure 29). Right clicking on the widget triggers a pop up menu with a list of textually similar widgets, windows and groups defined by their spatial organization (Figure 30). The icons show the type of design component represented. The first is a label, the second a spatial group, the next is a text area and the remainder are text fields. Navigating to the second item in the menu leads to a spatial grouping of push buttons within a window (Figure 31). Note that information about distance shows up in a single combo box in Figure 30 but appears spread across three different push buttons in another user's partial design (Figure 31).

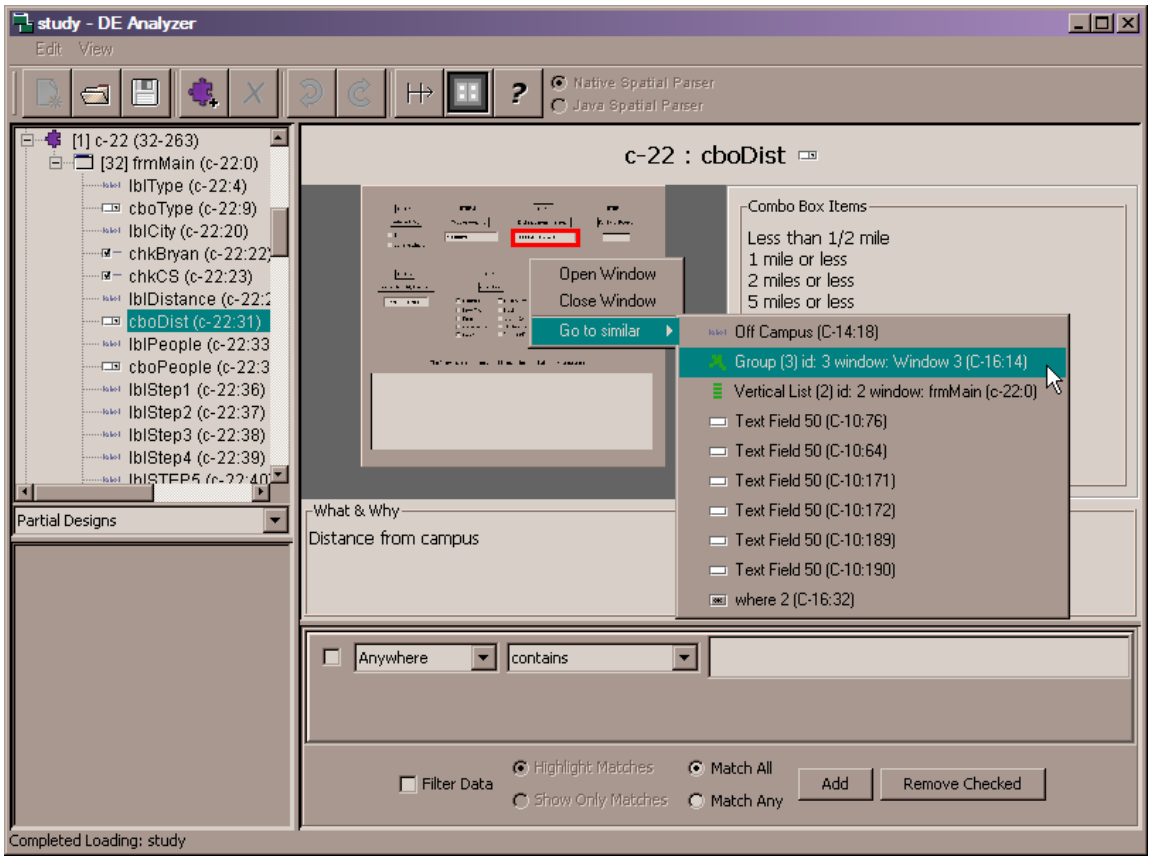


Figure 30. Right click navigation pop up menu to similar design components

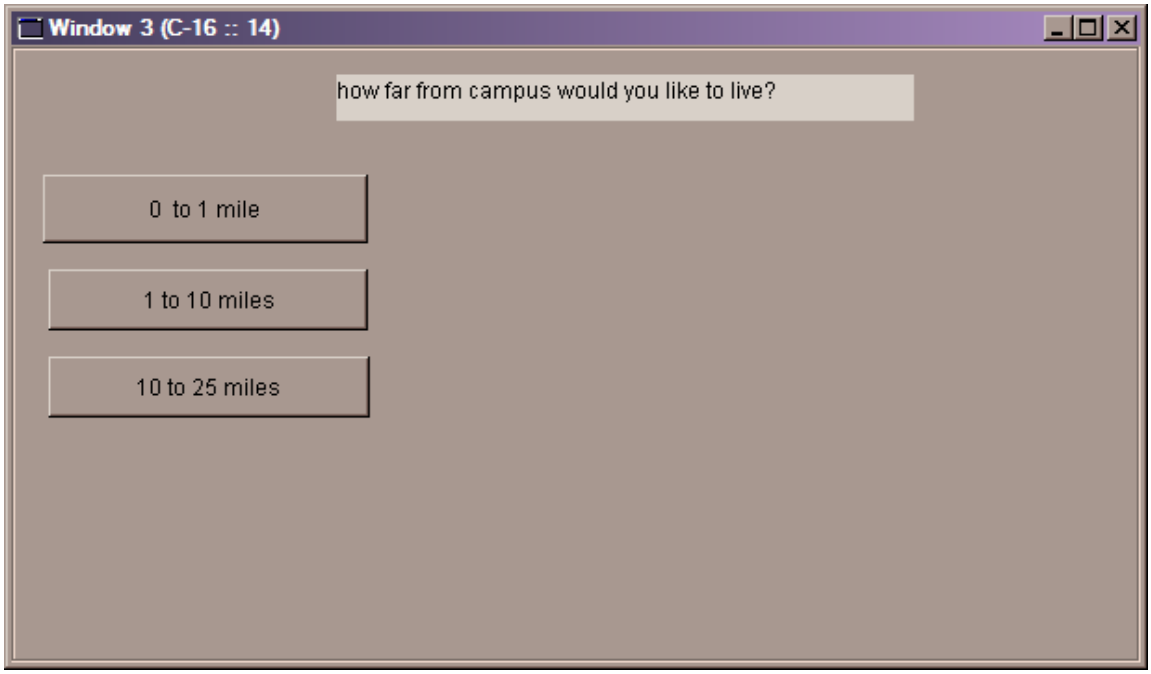


Figure 31. Distance information spread across three push buttons.

Next, the designer chooses to examine the term dictionary to identify common domain terms. The terms initially appear in alphabetic order. Each term's frequency within the set of partial designs is shown in parentheses. The designer modifies the view of the term dictionary to remove numbers and stop words from the list of terms. Next, the designer re-orders the terms according to frequency in descending order so that more frequent terms are at the top.

The designer sees that the term "apartment" occurs with a high frequency. The appearance of this term is not surprising based on the task. However, the designer uses this term to explore how "apartment" is used in various partial designs. When a particular widget under "apartment" is selected in the tree, a thumbnail of the window holding the widget along with any associated textual argumentation is displayed (Figure

32). Notice that the term “apartment” is highlighted by using blue bold text to assist locating the term within the widget’s display area. Also, the selected widget is highlighted with a red border in the thumbnail of the containing window.

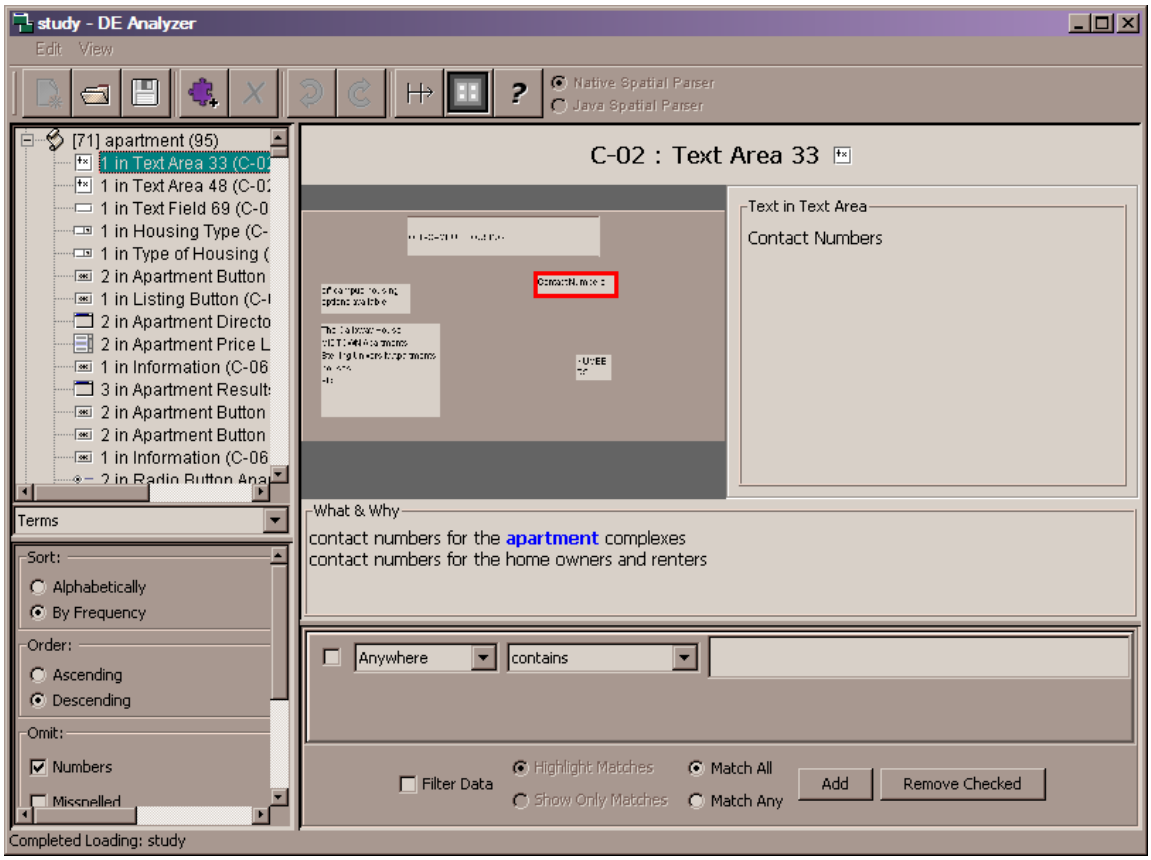


Figure 32. Widget selected under "housing" term.

The software designer comes to the term “bedroom” and thinks it would be interesting to see where the terms “bedroom” and “bath” occur together. So the designer types bath into the search overlay window and turns filtering on. Search results are projected onto the various views. Items containing a search term are highlighted with a

yellow background. Moreover, terms matching the search are highlighted with a yellow background in the widget's display area (Figure 33). An alternative to projecting the search results onto the tree is to "hide" any node in the tree that does not match the search criteria (Figure 34).

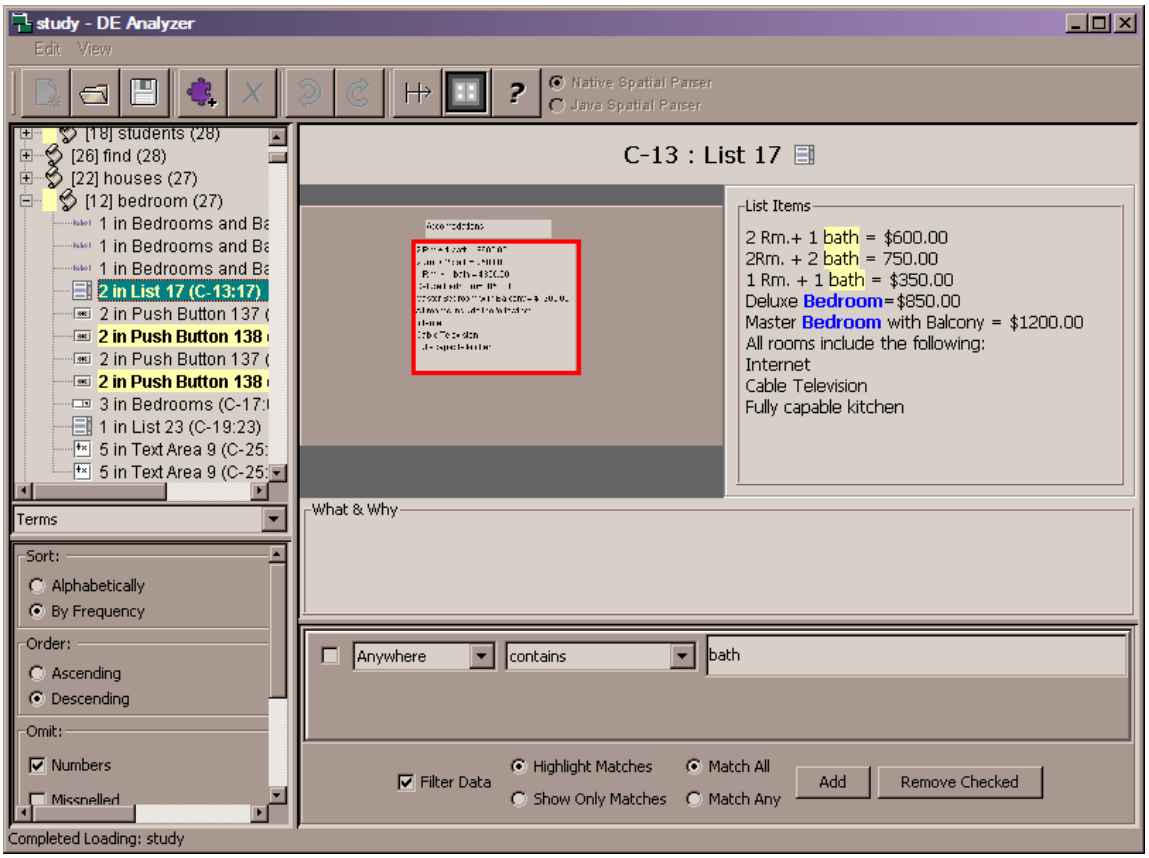


Figure 33. Search terms highlighted.

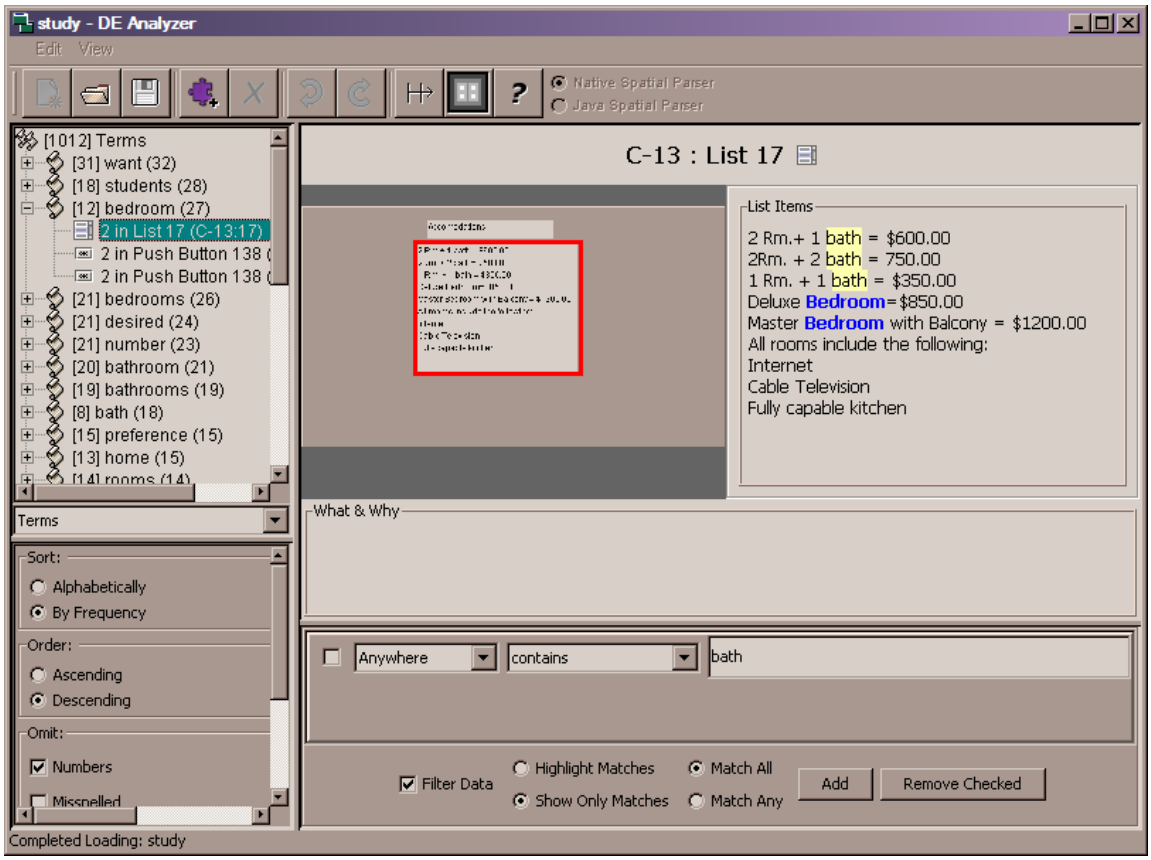


Figure 34. Search only results shown.

In the process of scrolling through the terms, the software designer ends up at the bottom of the list where infrequent terms are found. Some of those terms are misspelled words. So the view is modified again to hide misspelled terms identified by the integrated spelling checker. The designer notices that the term “grad” appears and looks to see how it is used. The designer finds that it is found within the phrase “Grad Student”, an item in a combo box identifying what year a student is. This is the only occurrence of “grad”, “freshman”, “sophomore”, “junior”, and “senior”. To examine the window more closely, the designer double clicks the widget where “grad” appears. This opens the window and allows for closer examination (Figure 35). Inspection reveals that

this window represents information for finding roommates. This indicates that finding roommates is a part of some college students' housing search task. The designer makes note to watch for additional aspects of this feature in other students' partial designs. Since only one user provided information about student classification, the designer will investigate whether this is desired and other students just did not think to include this in their partial designs.

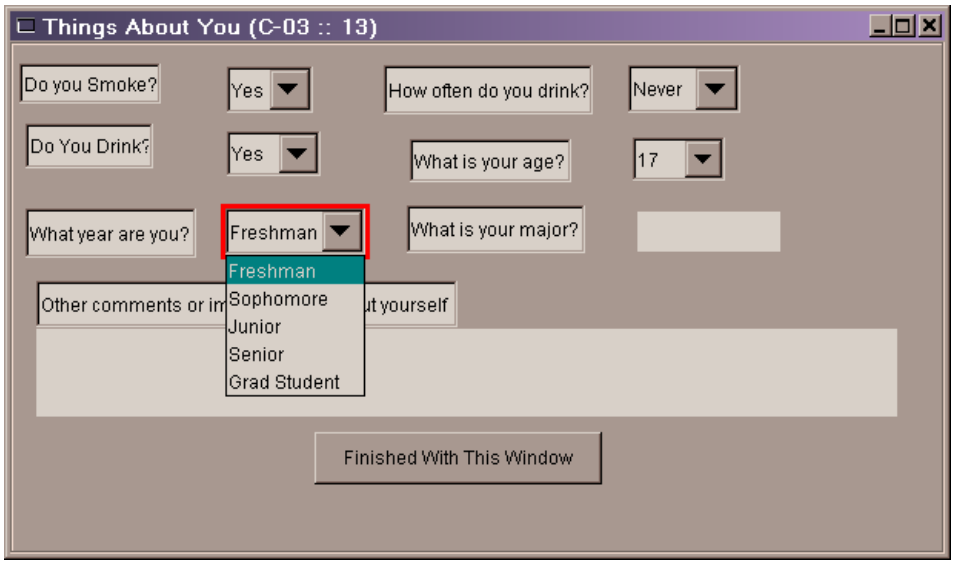


Figure 35. User design with the infrequent term "grad".

The above scenario shows examples of a software designer browsing through a collection of annotated designs based on terminology, design component and search. The next section describes the underlying computational support.

5.6 Summary

Designers' exploration of end user expressions is aided by the Design Exploration Analyzer tool. The back end provides computational support via term

vectors, spatial parsing, and clustering. This enables search overlay and similarity navigation between design elements created by end users. Designers see this information through the perspectives of partial designs as generated, terms appearing in designs, spatial groupings identified by parsing, and identified clusters.

6. EVALUATION: DESIGN EXPLORATION BUILDER

Evaluating the Design Exploration process requires collecting a set of user expressions and exploration by software designers. At the highest level, evaluation is for the entire process. At the same time interesting questions related to users and their expression of software desires and expression modalities can be assessed. Data collection covered the entire process and occurred in two phases. The first phase collected user expressions. At the same time users evaluated the Design Exploration Builder and the process. In the second phase software designers explored user expressions acquired during the first stage while providing feedback on the process and the Design Exploration Analyzer.

The overriding task used in the evaluation for both phases is creating software that helps college students find housing. In the first phase students are the domain experts and potential users of the system. In the second phase software designers explore user expressions generated for this task in order to identify domain information and software features.

> A software company is developing a software program to help college students find housing. Since college students will be using this program, you have been asked to provide input for the program.
>
> You are to provide as much information about what this program should do during the next hour. Please start with what you think is most important.

Figure 36. Task given to study participants.

6.1 Experimental Design

In this study 80 students from the undergraduate psychology subject pool provided information for the development of a software program that would help college students find housing. This study examines the information obtained through textual (25 participants), primarily graphical (27 participants) and combined approaches (28 participants). In the primarily graphical condition, users construct interface mockups, but cannot add additional textual argumentation although they can add widget related text such as button labels and list items. In the textual condition, users provided input using Microsoft Word.

6.1.1 Experimental Procedure

1. Participants provided demographics information.
2. Participants were given a quick tutorial on the Builder tool if in the primarily graphical or combined conditions.
3. Participants responded to the task given in Figure 36 with a time limit of one hour.
4. Participants responded to a follow up survey (Table III).
5. Participants were debriefed.

Table III. Task survey items

T1	I was able to describe everything I wanted to express.
T2	I had enough time to complete this task.
T3	I enjoyed doing this task.
T4	End users should be given opportunities like this to help design software.
T5[a]	Being able to create mock-ups of user interfaces would have been helpful.
T6[b]	Being able to type textual descriptions and explanations would have been helpful.
T7[b, c]	Providing information this way is better than using only text.
T8	I would like to provide information this way *in addition to* other avenues of input.
T9	I would like to provide information this way *in place of* other avenues of input.
T10	What, if any, tools would have helped you express your self better?
T11	What kind of information would you want to express that you cannot express with this approach?
T12	How would you improve this task?

a Text condition
b Primarily graphic condition
c Combined condition

6.2 Results and Discussion

Responses to items T1-T9 were collected using a scale where 1 was strongly disagree, 3 was neutral and 5 was strongly agree. These responses were evaluated using a single mean t-test against an expected value of 3.5 (Table IV). This expected value was chosen over the neutral value 3 since users tend respond with inflated values in an attempt to please investigators. In each condition, statements T1-T9 were also compared using paired t-tests (Table V).

Table IV. Average responses and (p-values)

Item	Text	Graphical	Combined
T1	**4.44** (<0.001)	3.67	3.79 (<0.1)
T2	**4.60** (<0.001)	3.70	3.32
T3	3.24	3.48	3.68
T4	**4.24** (<0.001)	**3.85** (<0.05)	**4.22** (<0.001)
T5[1]	**3.92** (<0.02)	NA	NA
T6[2]	NA	3.81	NA
T7[2,3]	NA	**4.04** (<0.01)	**4.11** (<0.01)
T8	3.68	**4.04** (<0.01)	**4.14** (<0.001)
T9	**2.80** (<0.001)	**2.63** (<0.001)	3.14 (<0.1)

Table V. Paired t-test for responses

Item	Textual vs. Graphical	Textual vs. Combined	Graphical vs. Combined
T1	**0.004**	**0.004**	0.649
T2	**0.006**	**0.000**	0.276
T3	0.348	0.110	0.408
T4	0.107	0.934	*0.070*
T7	NA	NA	0.781
T8	0.135	**0.040**	0.663
T9	0.433	0.166	**0.040**

Responses to the freeform questions (T10, T11 and T12) were categorized to help compare responses. Responses to T10 and T12 tended to be similar since subjects do not differentiate between tools that would be helpful and ways to improve the task itself. Only categories identified in at least 5 subjects in a condition are reported. For T10, 5 subjects in the graphical condition (19%) and 2 subjects in the combined

condition (7%) wanted the ability to explicitly link buttons created in the construction tool to other windows created. 10 subjects in the text condition (40%) and one subject in the combined condition (4%) wanted examples, sample programs, or the interface to other similar programs as a starting point or as material to critique.

Similarly in T12, 8 subjects in the text condition (32%) and 2 subjects in the graphical condition (7%) wanted material to start from, survey, or critique. 3 subjects in the graphical condition (11%) and 5 subjects in the combined condition (18%) wanted more direction on what they were supposed to do and what was expected from them. 4 subjects in the graphical condition (15%) and 6 subjects in the combined condition (21%) wanted more features in the construction tool such as snap to grid.

In T11, 2 subjects in the text condition (8%), 5 subjects in the graphical condition (19%) and 6 subjects in the combined condition (21%) wanted to express multimedia content such as images, video and audio. Finally, 6 subjects in the text condition (24%) wanted to express layout information.

A subset of the data was analyzed to get a feel for the types of information elicited in each condition and how the information compares across conditions. Subjects who said they enjoyed the task (responses of 4 or 5 on T3) were considered. All subjects that responded with 5 were used and subjects were randomly chosen from the remaining pool of subjects that responded with 4 to bring each group to 8. Relevant results will be introduced during discussion.

6.2.1 User Involvement

Users feel they should be given opportunities to help design software (T4). Since there is no statistical difference in the responses between groups, this is probably a general attitude unrelated to any of the conditions. Similarly, it is interesting to note that no group particularly liked their task although none significantly disliked it either (T3).

6.2.2 Communication

Both the graphical and combined groups would like to provide information as they did in addition to other methods and the combined was significantly different from responses by the text group (T8). However, having these tasks as the only way to communicate information was rejected by subjects in both the text and graphical conditions (T9). Note that the condition that allows both textual and graphical expression was not rejected. The only significant difference between groups was between the graphical and combined condition. Subjects in both the graphical and combined condition felt that providing information in their condition was better than if they were limited to using only text (T7). There was no significant difference in responses in these two conditions.

Subjects in the text condition were the only group to significantly say they described everything that they wanted to express and their response was significantly higher than both the graphical and combined groups (T1). They were also the only group to feel that they had enough time and significantly more so than the other two conditions (T2). But they also indicated that the ability to create interface mockups would be helpful (T5). Also, when asked about information they could not express (T11), 24%

indicated they could not express layout information. It could be that they finished their information more quickly and were bored as they waited for the time to expire since subjects were required to do the task for an entire hour. In any case, response rates to questionnaires can be low. The construction tasks may be more engaging and garner higher response rates.

An unexpected result was that subjects in the graphical condition did not feel that being able to add text would be helpful (T6). This is reinforced by the fact that in the combined condition, 9 of 25[1] subjects (36%) provided zero or one words of additional text that was not available to subjects in the graphic condition. Those individuals that did not provide additional text basically created interfaces under the same conditions as subjects in the graphical condition. This may explain why there was only one significant difference between responses in the graphical and combined conditions. The conditions may have been more similar than expected.

Subjects in both the graphical and combined conditions wanted abilities for further refining their interfaces mockups. These ranged from features assisting design, to the ability to show multimedia elements to being able to explicitly link widgets produced to each other. The interface construction tool was intentionally designed to prevent the creation of polished interfaces. If they spend most of their time refining the interface rather than providing additional information, it is feared that users might become married to their own designs. This would be acceptable only if that user was going to be

[1] 3 subject's task data was lost and could not be analyzed.

the only user. However, the goal here is to collect information from a broad segment of users to get a feel for the whole population.

It might be good to find a balance in the construction tools by adding additional support that eases the expression of semantic information in the communication task while still maintaining the feel of a low fidelity interface that cannot be polished.

Subjects in the graphical and combined conditions wanted more direction in the task. This might be due to being unfamiliar with the task, due to expecting their designs to be critiqued, or due to the potential of providing so many different levels of design. Users were not intended to get too much direction since that could have biased their responses. We wanted to see what they felt the design should include, not refinements for what we had in mind for the design. Similarly, those in the text condition indicated a desire for information to critique. These both probably deal with subjects trying to overcome a blank slate. This can be done by providing more information and examples. However, doing so can reduce the number of original ideas generated [Jansson and Smith, 1991]. Note that the desire for starting material and material to critique was practically non-existent in the two groups engaged in the construction task of creating interface mockups. So involving users in this way directed their focus to the task where they wanted more direction rather than a desire to see materials to critique.

6.2.3 User Representation

Domain characteristics identified in user experiments overlapped between all of the conditions, especially in the area of domain data. Frequently cited characteristics probably indicate items central to the domain. Items that are not cited frequently are also

interesting. These represent items that might be important yet get overlooked. Getting input from a broad range of users can provide a breadth of information that could easily be missed by a smaller set of representative users. There were a total of 47 domain characteristics identified across the subset of 8 participants from each condition. These characteristics are based on one investigator's analysis of the results. Most characteristics were identified by multiple subjects across all the conditions. However, 17 (36%) were identified by single subjects, and 8 more (17%) were identified by only 2 subjects. A few representative users would have probably missed many of these characteristics. These identified characteristics now add to the information gathered from participants.

6.2.4 Modes of Expression and Preferences

Some users have preferred modes of communication. Even when the mode of communication is constrained, people will find a way to use their preferred mode. One subject created a single page of output that looks surprisingly like a web page (Figure 37). This had the lowest textual volume of any subject in the text condition. This "text only" output incorporates the use of colored fonts (red and blue) and varying font sizes. The grey text at the bottom instructs readers to call one of two offices to get more information. This user's data was not analyzed for content in this paper.

Figure 37. Text subject forcing graphics.

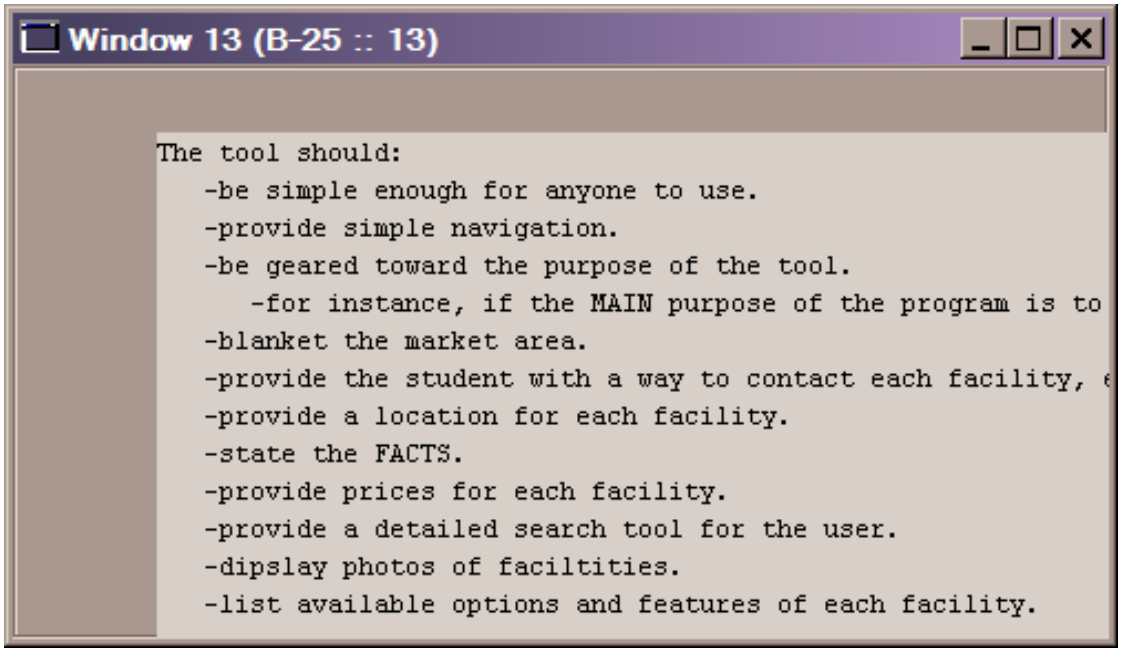


Figure 38. Widget used to provide description.

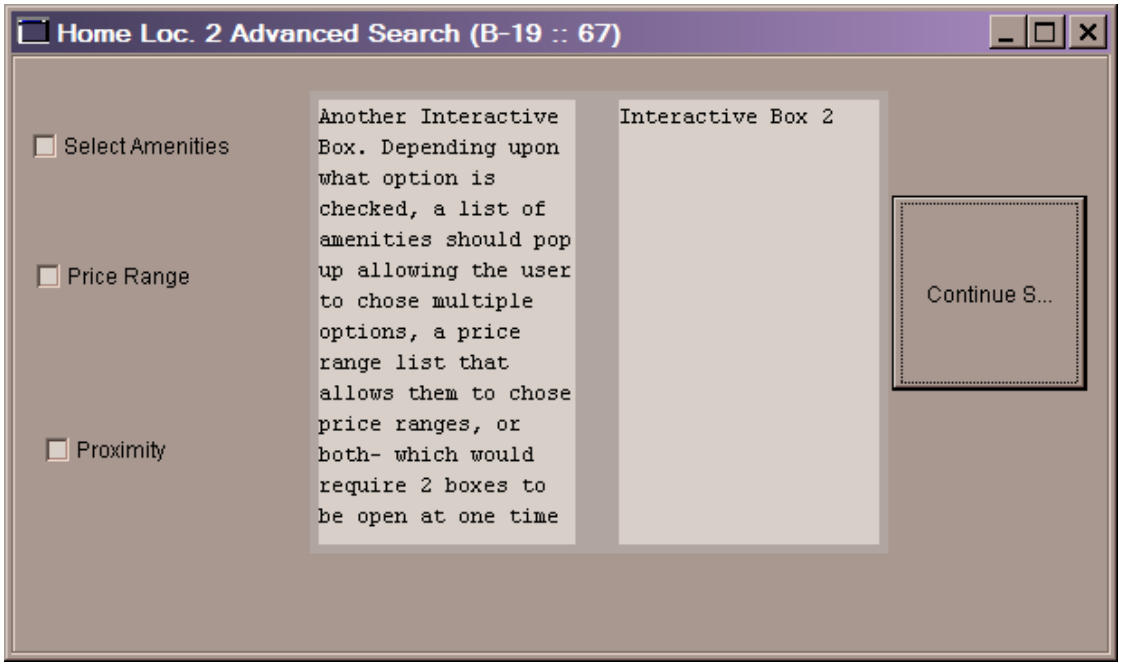


Figure 39. Interface incorporating description in text widget.

Users in the graphic condition found ways to provide textual descriptions. Recall that text could be added to widgets. The text area and text field widgets were used to provide textual descriptions. One user created a window with a text box for the sole purpose of providing some high level textual descriptions (Figure 38). Others incorporated descriptions into text widgets that were incorporated into the interface design (Figure 39). Other users found the generic widget which was intended to allow a place holder for a widget not provided and the description was to describe how the widget worked. One user used this to describe the program while not having that description be visible in the interface. Note the dark rectangle in Figure 40.

Some subjects even used these techniques when in the combined condition. Providing users options for mode of expression lets them communicate in the mode most comfortable for them. This behavior was also observed in a small prior study [Moore and Shipman 2000].

6.2.5 Types of Information

Data collected in all of the conditions provide a large amount of housing domain data that did not seem to differ significantly between groups. That is to say most groups identified the need for a search function based on various parameters and what those parameters should be. Moreover, domain characteristics were identified that cut across groups as discussed under ‘User Representation.’ However, there were some areas where the types and level of information differed.

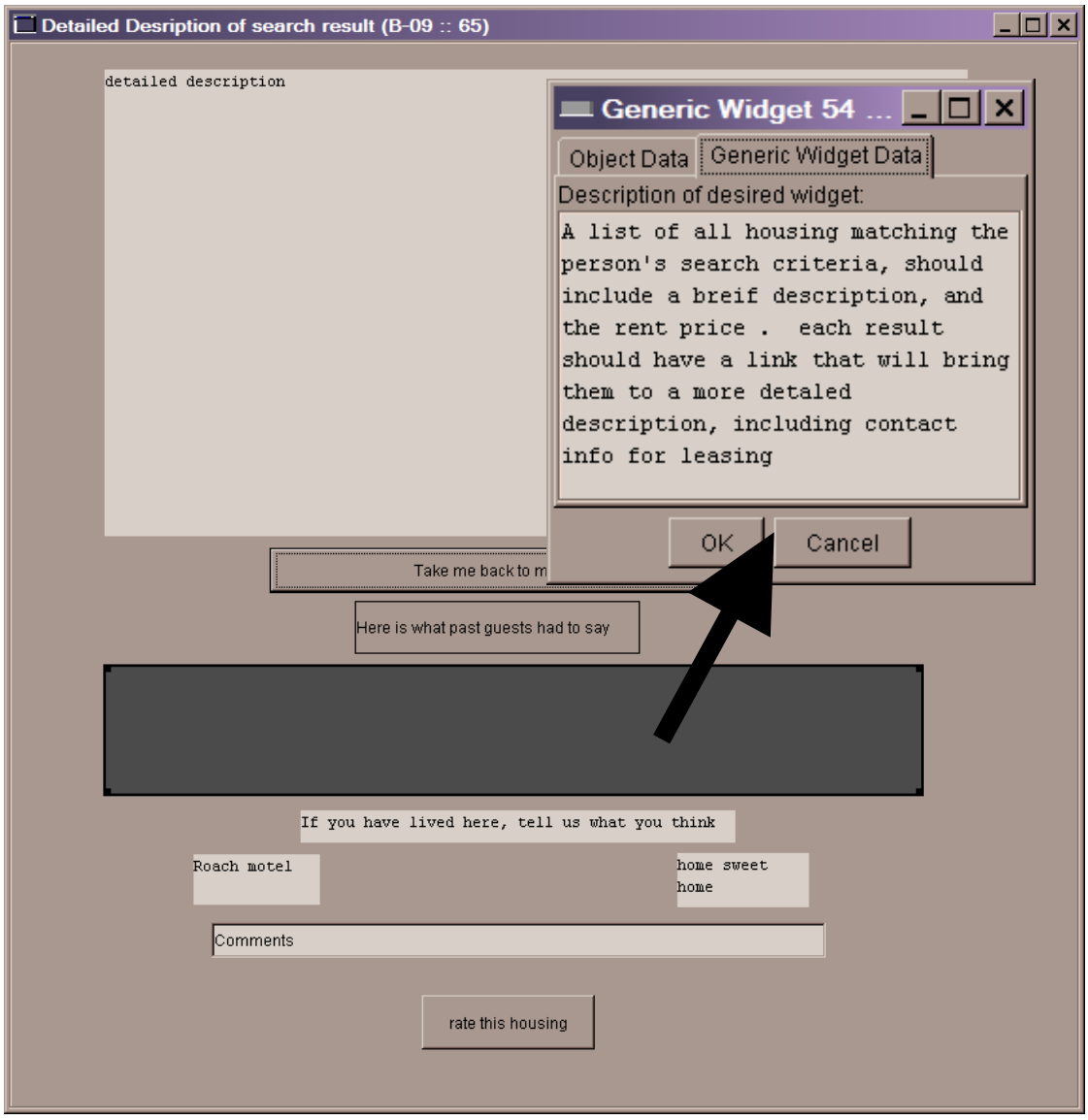


Figure 40. Interface with workaround for generic widget.

Users in the text condition differed in the level of information provided. Subjects stated they wanted the program to be easy to use, easy to understand, easy to follow, and easy to navigate. All of these comments came from subjects in the text condition. Of course what is “easy” is open to considerable debate. Often the designed, easy to use system does not meet user expectations and thus fails. Getting at what users consider

easy is important. Users in the other two conditions don't explicitly say the program should be easy, instead they provide interface examples that implicitly represent what they have in their mind.

Text subjects also provided 36 of 40 (90%) items that were categorized as high level requirements and requirements about the system as a whole that did not relate to interactions with the program. In the same way, many of these requirements were not made explicitly by users in the other conditions, but were indicated through their interface constructions. For example one text subject stated that each housing type had its own criteria. This same concept was expressed by other subjects by designing separate interfaces for searching and viewing results depending on the type of housing. Another text subject wanted "Small information selections that provide the user with just enough basic information to choose to visit specific pages." Subjects in the other conditions showed examples of this. One user provided a rectangular object (Figure 40) and in a related description indicates this should be a short description and the rent price. Other users showed this through the interface design (Figure 41 and Figure 42).

Figure 41. Results page as a table.

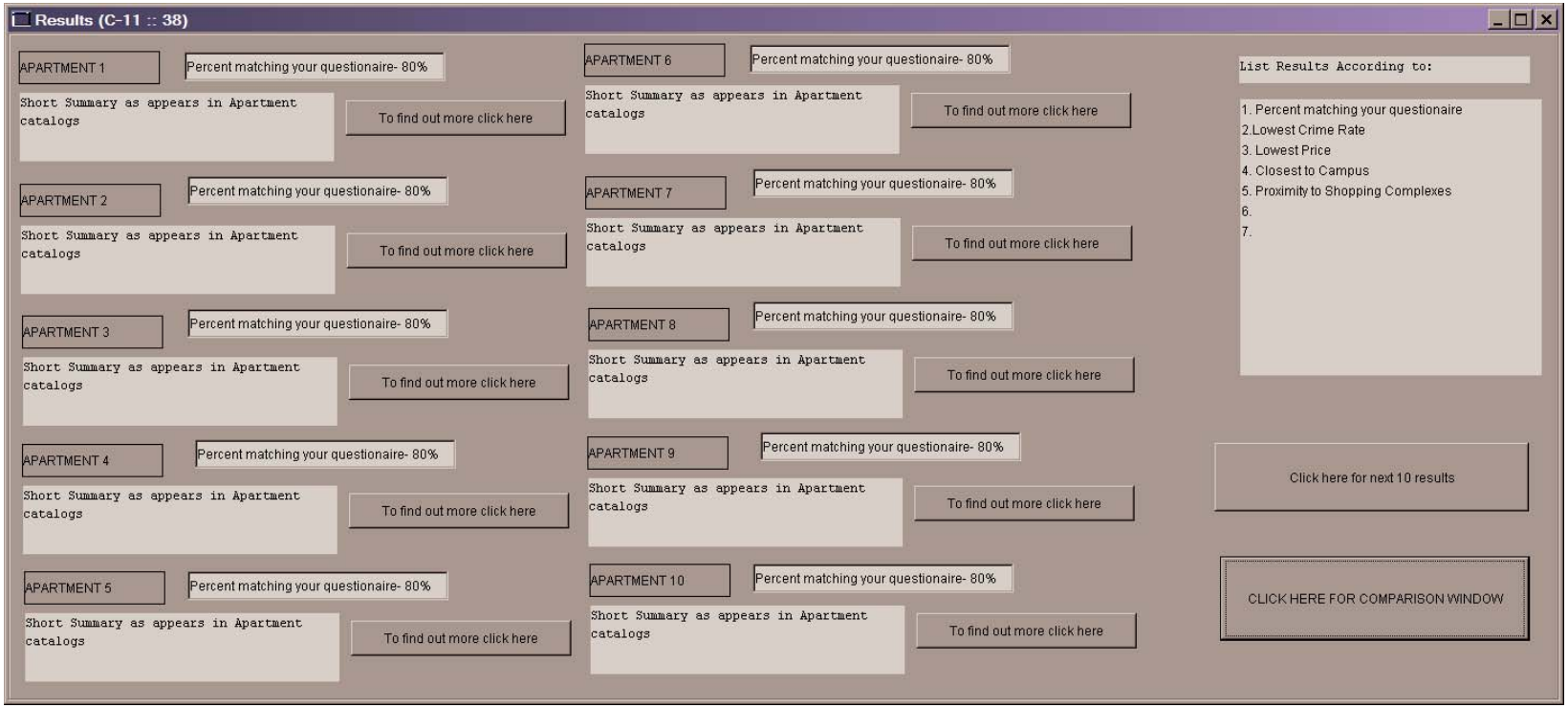
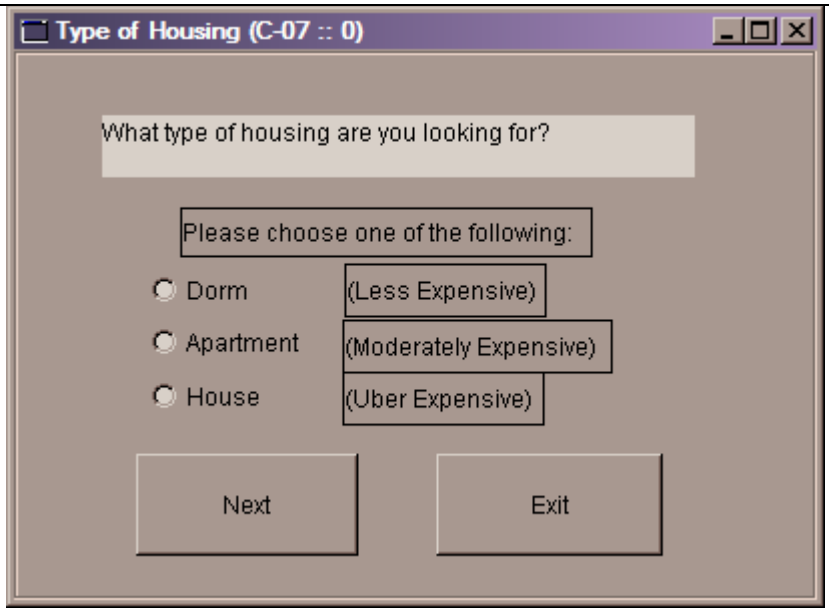


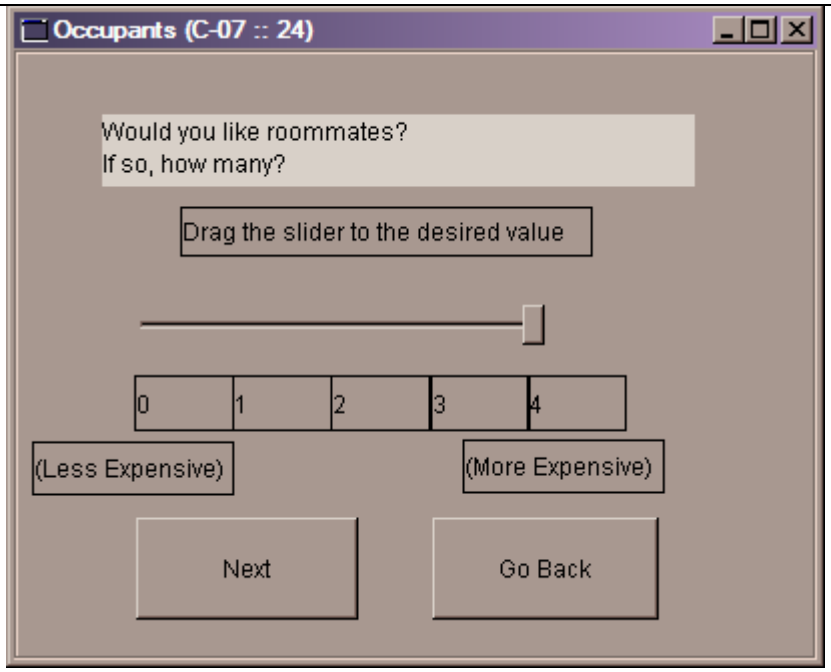
Figure 42. Results page incorporating grouping.

Users in all conditions communicated preferred interaction paradigms. One text subject wants each screen to provide access to the search engine for simple questions. Other users illustrated ways of interacting through their interface constructions. Figure 42 shows a sorting behavior according to items in a list. Some users created interfaces that took users through multiple windows when defining the criteria for a search (Figure 43) while others provided a single window with many search criteria on the window (Figure 44 and Figure 45).

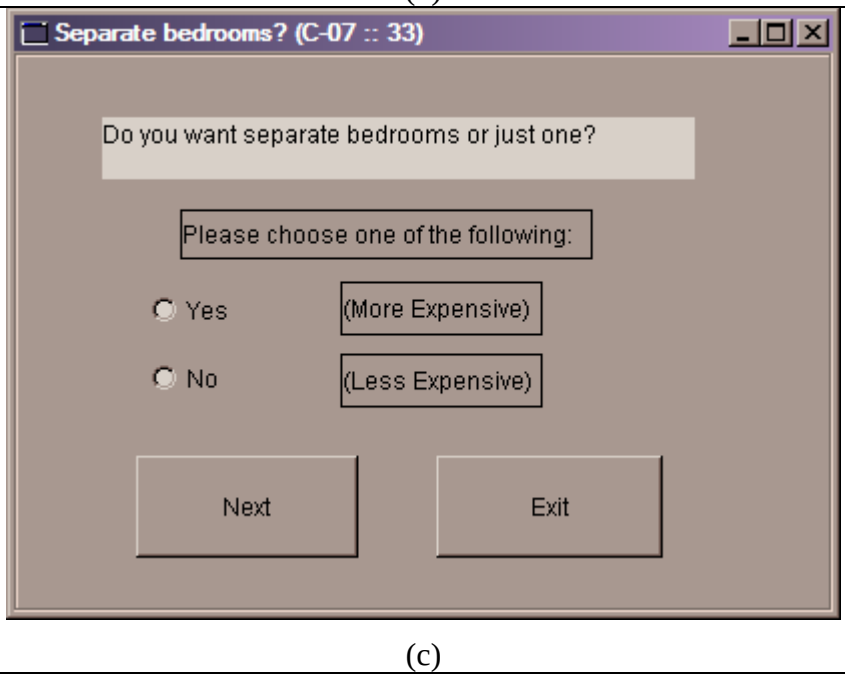


(a)

Figure 43. Multiple windows defining search criteria.

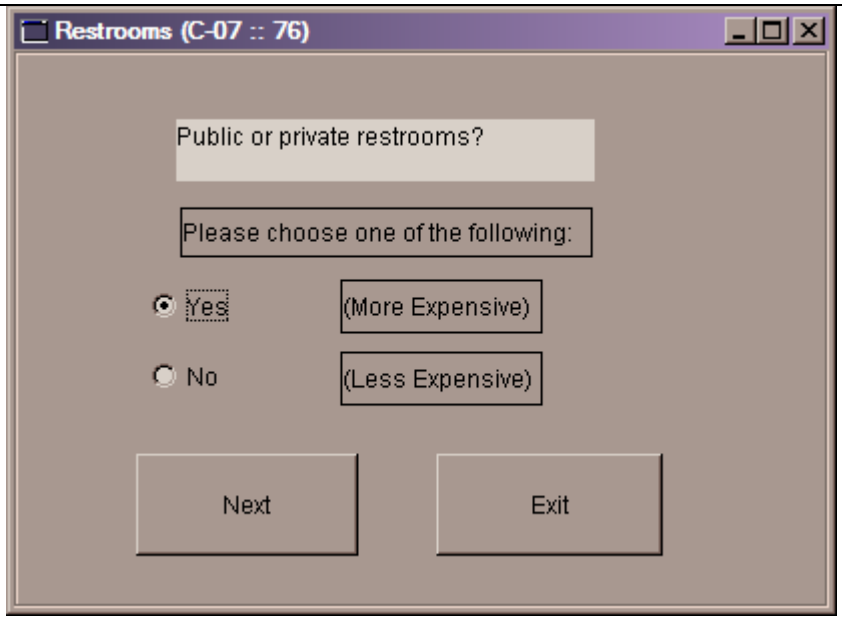


(b)

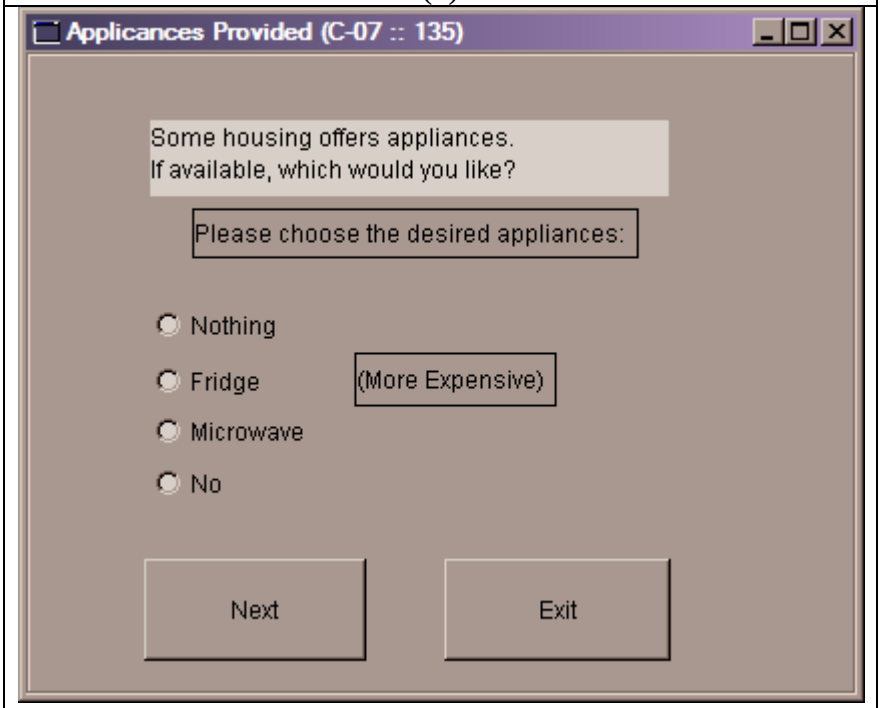


(c)

Figure 43. Continued.



(d)



(e)

Figure 43. Continued.

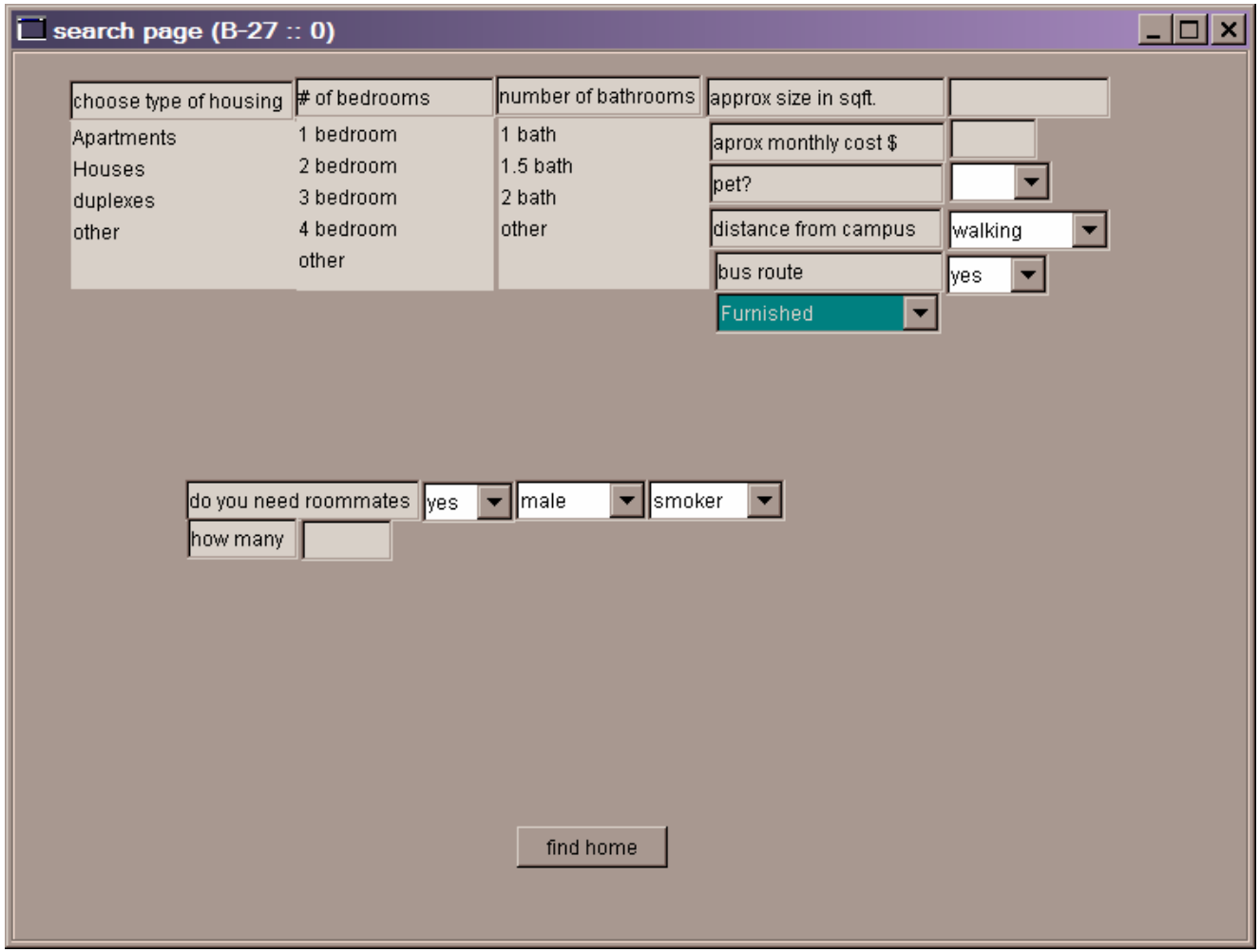


Figure 44. Interface incorporating many search items in one window.

Figure 45. Another interface incorporating many search items in one window.

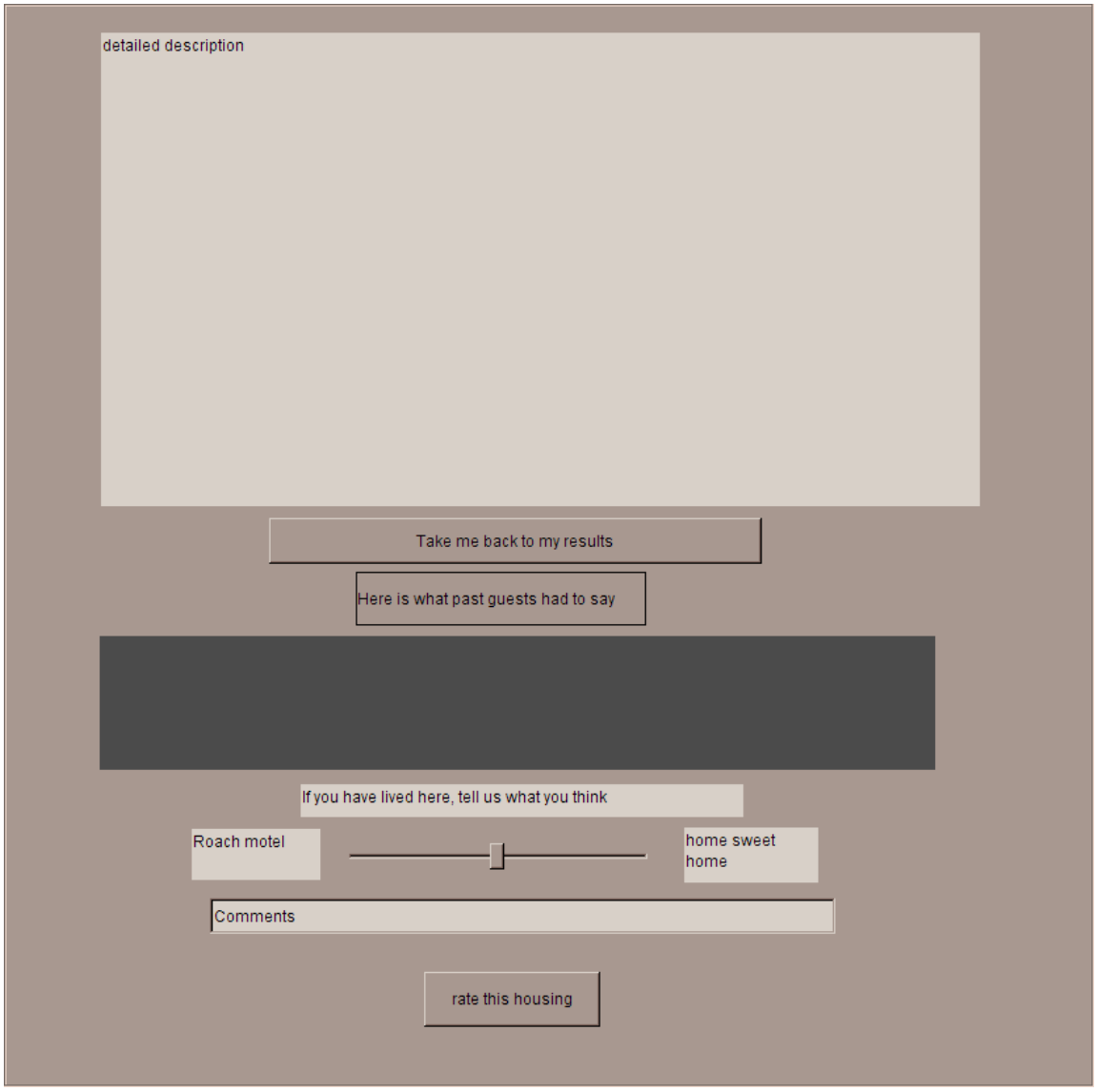


Figure 46. Scale with interesting end values.

Some users used text fields to input search criteria where others used selection via a drop-down list or radio buttons. Looking at the frequency of use of each of these approaches in a population gives insight into how the population as a whole wants to interact.

Finally, the labels people use give insight into the types of labeling they prefer. For example relative to distances from campus many users created scales based on number of miles. Others abstracted this distance into descriptive labels such as “closest to campus,” “middle distance,” and “farthest from campus.” Another subject used “near,” “far,” and “doesn’t matter.” Other subjects reported the distance relative to time in minutes. One user combined time with references to travel activity (“walking,” “biking,” “10 min,” “15 min,” and “any distance”). Figure 46 shows an example of a rating scale that goes from “roach motel” to “home sweet home.” Subjects in the text condition did not provide these types of insights. They emerged in the graphical and combined groups when they were engaged in a task that more closely matches the design of the system. While this information could be solicited from users after initial requirements and domain modeling, the Design Exploration process elicited this information without additional designer intervention.

6.3 Summary

Even though a goal of Design Exploration is freedom of expression, the conditions of expression constrained users. Those communicating with interface construction wanted to convey additional types of content such as images and expressing semantic content more directly. In both the textual and graphical users found ways to

work around expression constraints. Most users seemed to deal with a blank slate syndrome. Interestingly, users in the textual condition wanted examples to respond to while those involved in constructing an interface mock up wanted more direction for their task. Finally, the types of information varied. In some cases interface designs indicated higher level requirements implicitly by providing an implementation that included more fine grained information.

7. EVALUATION: DESIGN EXPLORATION ANALYZER

The Design Exploration process involves the collection of information from end users and the exploration of that information by software designers. The second phase of the study uses data collected in the first phase, i.e. the Builder study. This second study allows a more summative view of the Design Exploration process. In the second phase of the evaluation software designers explore the user expressions collected in the first phase of the Design Exploration evaluation.

7.1 Experimental Design

In this study fifteen advanced graduate students from a computer science department with classroom and practical experience in system design take the role of software designers in the Design Exploration process. Ten of these designers explored user expressions. Of these, five participants were given access to user expressions collected in the primarily textual condition. The other five were given access to user expressions collected in the combined graphical and textual condition. The remaining five participants function as a control group and answer many of the same questions based solely on their own knowledge of the domain and system design (i.e., without access to user expression from the first phase).

Both groups with access to user feedback from the first phase explored user expressions in the Design Exploration Analyzer. However, the Analyzer was modified for the primarily textual condition. For this one group, all user text was incorporated into a single text area widget to allow use of Design Exploration Analyzer text tools. Also, to

Table VI. Amenity and main task questions

Item	Design Exploration & Text	Control
AT	Spend fifteen (15) minutes identifying amenities for housing as expressed by users. No rationale required.	Spend fifteen (15) minutes expressing amenities for housing. No rationale required.
MT1	Based on user input, what are the 4 most popular features indicated (other than search)? Please provide rationale.	What do you think users would consider the 4 most popular features (other than search)? Please provide rationale.
MT2	In your opinion, what are the 4 most important features (other than search)? Please provide rationale.	In your opinion, what are the 4 most important features (other than search)? Please provide rationale.
MT3	What important features were identified by only one or a few users? Please provide rationale.	
MT4	What features are needed that are not mentioned by any users? Please provide rationale.	
MT5	Describe the interaction style most users would want in the application and how you came to that conclusion. Please provide rationale.	Describe the interaction style most users would want in the application and how you came to that conclusion. Please provide rationale.
MT6	Which users did you find most valuable? Please give rationale for each user mentioned.	
MT7	Which users did you find least valuable? Please give rationale for each user mentioned.	
MT8	What additional information needs to be requested from users? Please provide rationale.	What information needs to be requested from users? Please provide rationale.
MT9	Share interesting points for software development that are not addressed in the preceding sections. Please provide rationale.	Share interesting points for software development that are not addressed in the preceding sections. Please provide rationale.

AT – Amenities Task
MT – Main Task

view user expression in its original context, opening a widget or window in the text only condition opens the document in MS Word as opposed to opening a user created window.

7.1.1 Procedure

The session proceeded as follows:

1. Tutorial: Participants working with the Design Exploration Analyzer started with an un-timed tutorial to learn their respective versions of the tool (Appendix B).
2. Scenario Introduction: All Participants were given a handout (Appendix C) and given a verbal introduction to the scenario.
3. Amenities Task: All participants were shown the results of a Google definition search for the word “amenities” (Appendix D). This was to ensure participants who might not be familiar with the term could quickly see what the term meant and spend time doing the task rather than understanding the term. At this point participants spent fifteen minutes listing amenities for housing. Table VI shows the wording of the amenities task question that was presented to participants for each condition. Participants in the control group came up with amenities on their own (i.e. without user input) while the other two groups searched user expressions presented in the Analyzer. Participants could stop when they felt they were done. This activity was separated from the main task to ensure this question was answered and not inadvertently omitted.
4. Main Task: Each group was given different instructions based on their condition (Table VII). Participants were given 45 minutes to answer questions for this more

involved task (Table VI). Since it was apparent that search was the main feature of this application, participants were asked to identify features other than search in questions MT1 and MT2.

Table VII. Main task instructions

Design Exploration	Text	Control
Spend the next 45 minutes responding to the following. The time constraint will not allow you to look through all users' partial designs individually and provide the required information. It is important to provide responses to everything in the given time. Use the provided features to navigate through partial designs to answer all of the following.	Spend the next 45 minutes responding to the following. The time constraint will not allow you to look through all users' documents individually and provide the required information. It is important to provide responses to everything in the given time. Use the provided features to navigate through documents to answer all of the following.	Spend the next 45 minutes responding to the following. It is important to provide responses to everything in the given time.

5. Demographic Information, Tool Survey, and Interview: Demographic information was collected from all participants. Further, those participants using a version of the Analyzer were given a follow up survey to evaluate both the tool and the process of collecting information from users in the manner explained in their scenario (Appendix C). Responses to the first 9 questions were collected using a scale ranging from 1 "strongly agree" to 4 "neutral" to 7 "strongly disagree." The last four

Table VIII. Follow up survey quantitative questions

Item	Design Exploration	Text
FS1	I identified useful domain information.	I identified useful domain information.
FS2	I could effectively navigate through the annotated partial designs.	I could effectively navigate through the textual descriptions.
FS3	I was able to infer users' desires expressed through their annotated partial designs.	I was able to infer users' desires expressed through their textual descriptions.
FS4	User annotated partial designs added to my understanding of the domain.	User textual descriptions added to my understanding of the domain.
FS5	Having more potential user annotated partial designs would make analysis significantly more difficult.	Having more potential user textual descriptions would make analysis significantly more difficult.
FS6	Having more potential user annotated partial designs would significantly increase analysis time.	Having more potential user textual descriptions would significantly increase analysis time.
FS7	A tool is needed to assist this process.	A tool is needed to assist this process.
FS8	The tool assisted this process.	The tool assisted this process.
FS9	Users creating annotated partial designs is a valuable avenue for collecting information from users.	Freeform textual descriptions is a valuable avenue for collecting information from users.
	How helpful were each of the following features?	How helpful were each of the following features?
FH1	Search Overlay	Search Overlay
FH2	Dictionary	Terms
FH3	Clusters	
FH4	Similarity Navigation	

FS – Follow Up Survey
FH – Feature Helpfulness

evaluated helpfulness of specific tool features. These were collected using a scale ranging from 1 “not helpful” to 4 “neutral” to 7 “very helpful.” Moreover, participants were asked open ended questions regarding the process and tool that they used. The wording for both the Design Exploration and text condition were identical (Table IX). Optionally, participants were interviewed to get clarifications that the interviewer deemed needed.

Table IX. Follow up survey open ended questions

Item	Design Exploration & Text
FQ1	What was good about the tool?
FQ2	What was good about the process?
FQ3	What was bad about the tool?
FQ4	What was bad about the process?
FQ5	What additional features are needed in the tool to assist this process?
FQ6	What were obstacles to understanding end user communication?

7.2 Amenities Identification

One goal of the evaluation was to assess end user expressions identified by software designers in each condition. One aspect is to look at the amount of information generated as well as the uniqueness of the information garnered. Fluency and novelty creativity metrics can help assess these aspects [Shah et al. 2003]. In this study these metrics are applied to each condition as a whole rather than to individuals. While it would be appropriate to apply these metrics individually to participants in the control

group, in the other two conditions participants are reporting information they find in the set of information as the whole. It would not make sense to attribute information that they identify to a particular participant.

In the study, software designer participants were asked to focus on a subset of the data. This bounds the scope of analysis to a set of information that is comparable and these short descriptions were also easier to analyze objectively. Amenity identification was used for this task. Originally, this task was included with the main task items. However, a pilot participant revealed time management issues. Consequently, this task was separated into an individual 15 minute activity performed prior to answering other questions in the software designer's exploration task. The control group represents a baseline for comparing what software designers identify from end user expressions with what a software designer would identify without this input.

7.2.1 Results

The amenity lists provided by participants were sorted and grouped together to form a set of concepts to compare the amenities produced in each condition (Appendix E). The grouping was done at a conceptual level. So, distinct terms such as "hot tub" and "Jacuzzi" were both counted as the same concept. Some participants' lists contained amenities that combined concepts identified by other participants as separate items. These items were split. For example, one subject identified paid internet. This was counted for both the concept of providing internet as well as under the concept of items included at no additional charge. Also, some amenities were considered as being part of a larger concept. For example, parking was identified by several participants while some

also indicated types of parking e.g., covered or garage. In some cases this resulted in taking what was presented as a single concept and analyzing it as multiple concepts. For example, one user responded “parking (street, lot, numbered, covered, garage).” In this case not only was the concept parking used, but also each type of parking identified was considered a separate concept.

7.2.1.1 Fluency

Fluency indicates “how prolific one is in generating ideas” [Shah et al. 2003]. For this analysis, fluency is reported as the ratio of concepts identified in a condition to the sum of concepts identified by all conditions. Several users in the control group identified the same amenities and used very similar phrasings. For example 3 of the 5 Control participants identified ceramic tile and no other type of flooring. It turns out that the Google definitions page shown to all users to ensure they knew the meaning of the term “amenities” also included examples that listed several amenities (Appendix D). This also explains why an amenity such as “hair dryer” appeared on this list since this is more commonly seen as an amenity for a hotel room and not one for residential housing. So a modified analysis (with correction) did not credit participants in the control group for concepts appearing on the definitions page. Participants in the Design Exploration and Text conditions were still given credit for those concepts since they were reporting what was found in end user expression, and those end users did not have access to this definition information when they generated their expressions. Both the corrected and uncorrected scores are presented since some users in the Control group probably

duplicated amenities on the definitions page without utilizing the definitions that appeared there. The corrected and uncorrected results are shown in Figure 47.

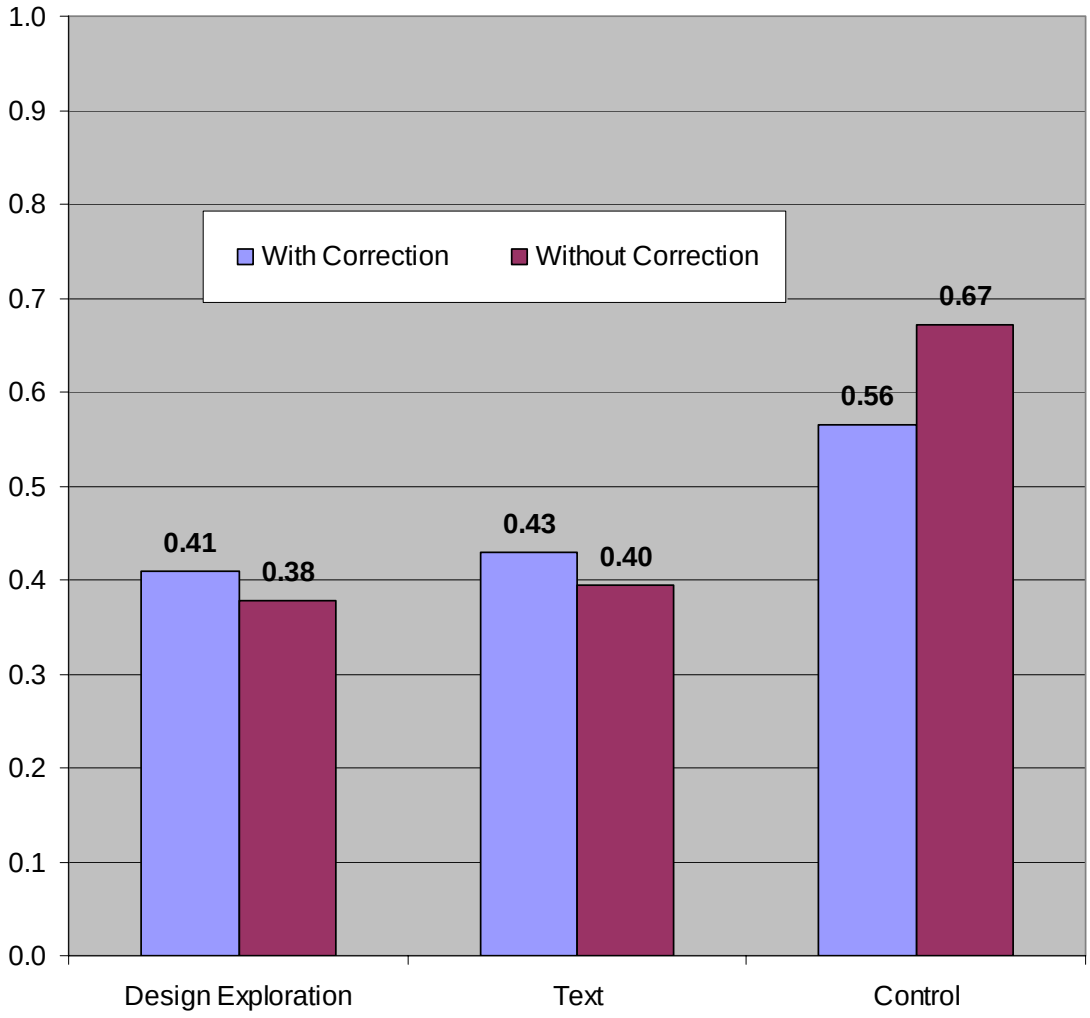


Figure 47. Fluency for amenities task.

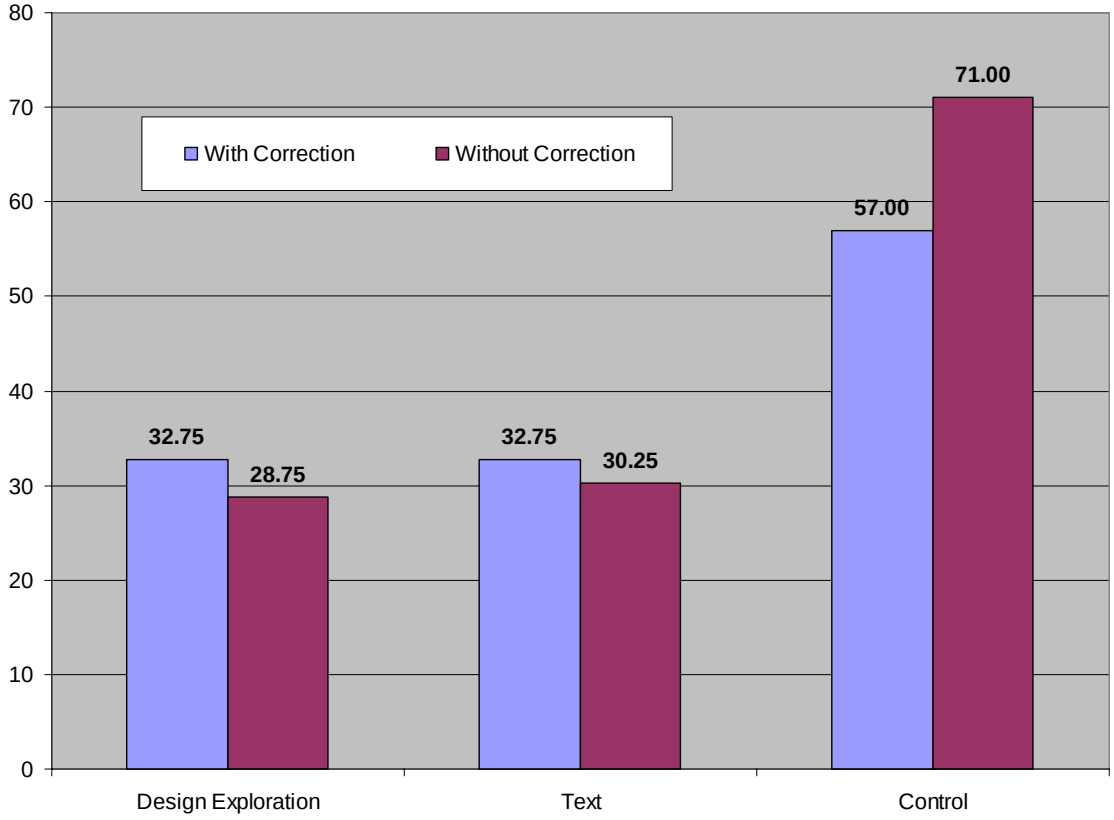


Figure 48. Novelty scores for amenities task.

7.2.1.2 Novelty

Novelty deals with the uniqueness of a concept in relation to others [Shah et al. 2003]. Concepts created by only one condition are considered highly original whereas concepts generated by all conditions are not original. Each concept identified in the study was assigned a novelty value of 1 (identified in only one condition), 0.25 (shared by two conditions), or 0 (shared by all conditions). A novelty score was calculated for each condition by summing the novelty values for each concept identified in that

condition. This score was calculated for both corrected and uncorrected cases as was described in the previous section. These results are shown in Figure 48.

It is also informative to look at a Venn diagram of the concepts identified by each condition to see the distribution of unique concepts, shared concepts, and concepts identified in all three conditions. The corrected results of the 154 amenities concepts are shown in Figure 49, and the uncorrected results for the 167 amenities concepts are shown in Figure 50.

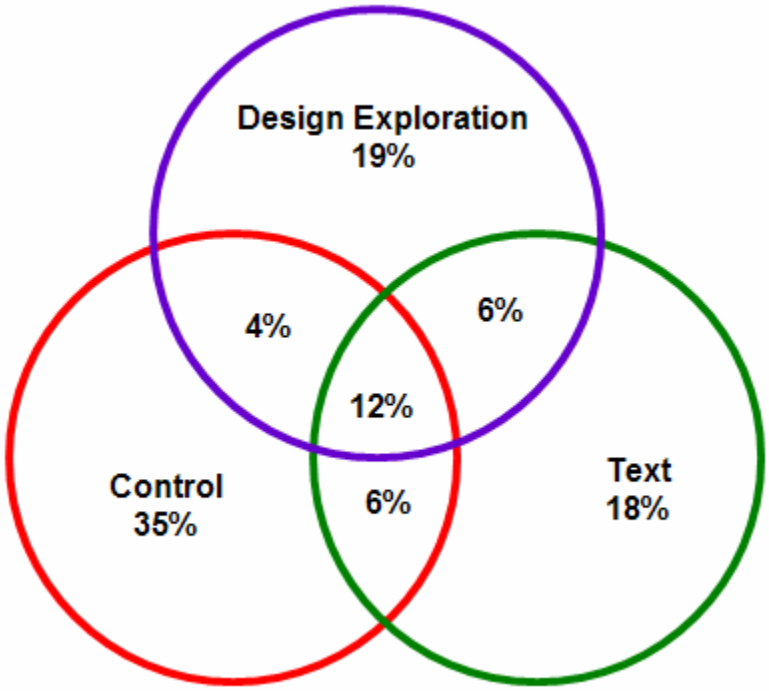


Figure 49. Venn diagram of 154 amenities concepts with correction.

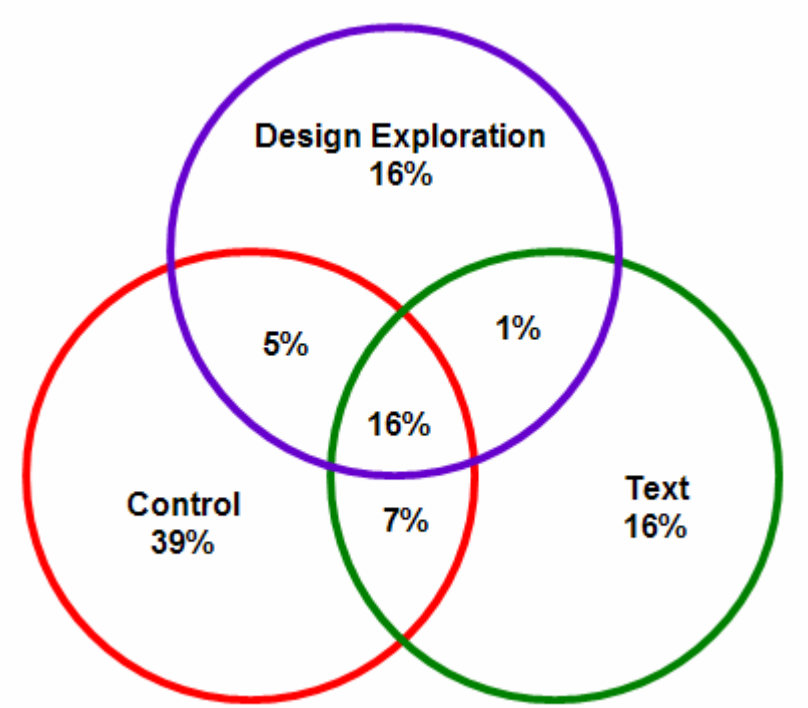


Figure 50. Venn diagram of 167 amenities concepts without correction.

7.2.2 Discussion

The expectation was that the control group would be less fluent and have a lower novelty score than both conditions gathering input from end users. The results do not support this. There are several factors confounding the results. The first was the influence of the definition page on the amenities identified as discussed before. However, even after correction the control group still had higher scores than either of the user groups.

Another issue is that all participants acting as designers were graduate students. Consequently, they are also members of the end user population. One of the goals of the

Design Exploration process is to collect domain information from domain experts for software designers who are not also experts in the domain.

Next, the nature of the task also contributes to these differences. End users in the Builder study were not instructed to focus on amenities specifically. Any amenities expressed by end users (and then identified by software designers) were generated as a side effect of their design task. Those designers in the control group were given a specific amenities task and were also seeded with amenities concepts through the definitions page. While some correction was made for specific amenities appearing on the page, other amenities sparked by larger descriptions were not addressed.

Finally, since this was an identification task for all except the control group, amenities could have been missed and in fact some were. For example, only one participant in the control group identified the type of housing as an amenity (specifically house, duplex, fourplex, and apartments). However, an even more extensive list was expressed by a user in an interface design (Figure 51). This accounted for four original concepts for the control group but was not identified by anyone with access to this user expression.

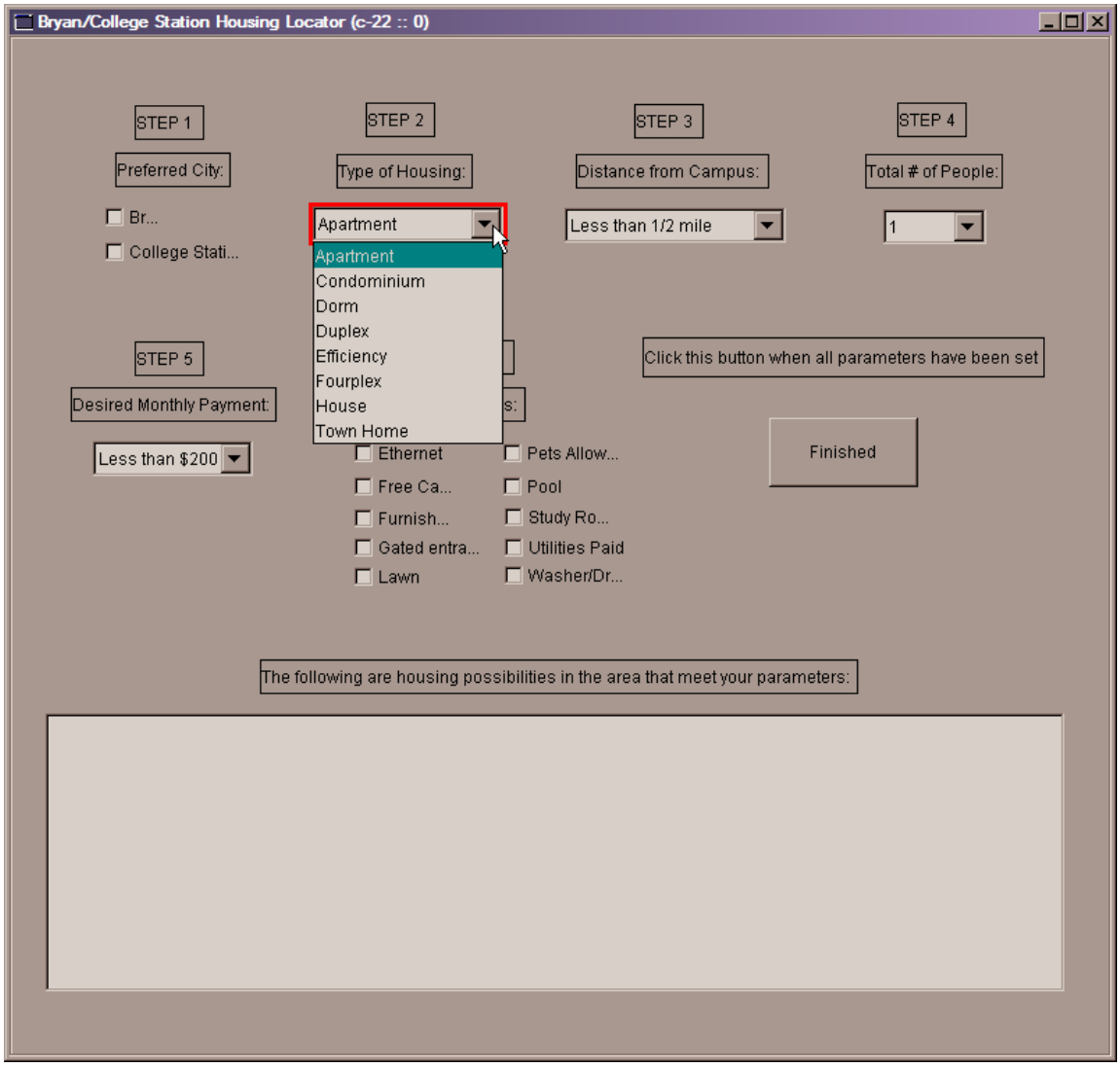


Figure 51. Combo box identifying types of housing.

One of the goals for including a large number of users is to gather a more complete set of software information than is available with a few. If a few users are able to provide this complete set of information, then the information gathered from each condition should be somewhat homogenous. Regardless and perhaps in spite of the problems encountered, the data indicates that approximately 70% of the concepts generated were unique to one of the conditions and only about 30% of the concepts were

shared by more than one condition. While it is unlikely that Design Exploration collects all of the information needed for software design, this data shows that including more end users results in the collection of a greater portion of this information than is possible with just a few users. This underscores that no single approach or person identifies all possibly relevant information. Moreover, the concepts generated through the Design Exploration process were roughly equivalent to those produced in the text condition.

7.3 Feature Identification

Software design involves understanding potential work practices and including features to support those practices. In the amenities task, the goal was to see fluency and novelty within the target domain so the scope of the identification was limited. However, features and their identification are harder to pin down. The goal in this task is to identify software features and priorities for those features as indicated by end users and designers. Fluency and novelty metrics do not apply to this analysis since responses were limited in number. However, comparisons of features identified in each condition and their overlap is informative.

7.3.1 Results

Each designer participant identified the four most popular features identified by users (or what Control group designers thought would be the four most popular features). Designers were then asked to list the four most important features based on their own opinions. The features identified in each case were organized and sorted in a similar way to amenities in the previous section (Appendix F). Figure 52 shows the

Venn diagram of the 24 popular features identified in each condition. Figure 53 shows the Venn diagram of the 24 important features identified in each condition.

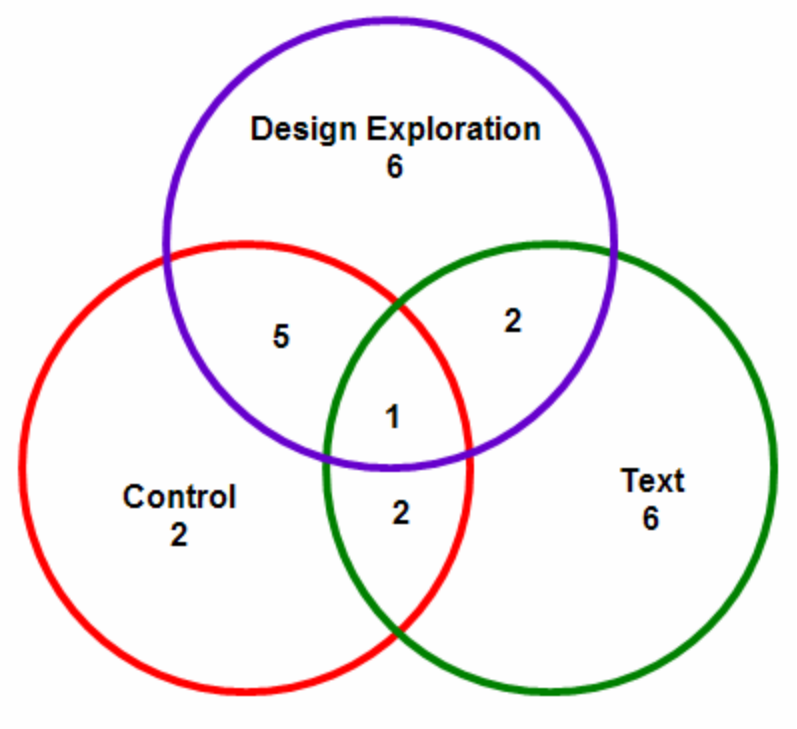


Figure 52. Venn diagram of popular features in each condition.

Participants (except for those in the control group) were asked to identify features indicated by only one or a few users, i.e. rare features. Some features were identified as both popular and rare, and in two cases a participant did this seemingly contradictory assignment. When there was contradiction with a user, the rare case was used since this response was elicited further in the feedback form. The use of the word "popular" might have led some participants to add an element of judgment so that they indicated what they thought would be the most popular features from those identified in

the data regardless of its frequency. Moreover, the second question was stated in a way that was more directly about frequency of occurrence. So these interpretation issues might explain these seemingly contradictory assignments. In three cases features were rare in one condition and popular in the other. In these cases, the features were allocated as popular rather than rare since it was popular in another condition. Alternatively, those identifying it as rare may have just overlooked additional occurrences when exploring the information under a time constraint.

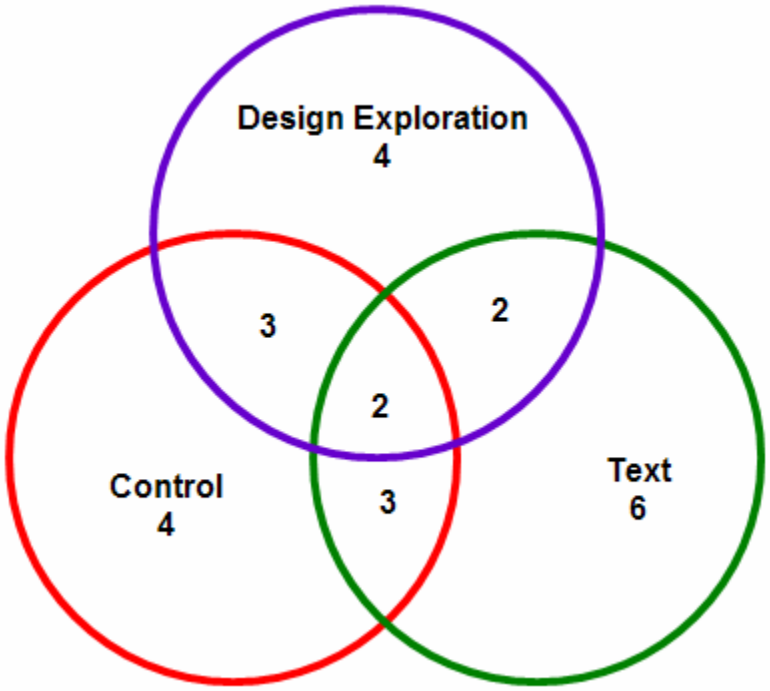


Figure 53. Venn diagram of important features in each condition.

The features identified as popular and the features identified as rare were compared with those identified as important too see what features considered important

by software designers were also identified as popular and rare. 11 popular features also appeared as an important feature. Furthermore, participants in the two groups pulling from user expressions identified 14 features that were only identified by one or a few users. 6 of these rare features also appeared as an important feature.

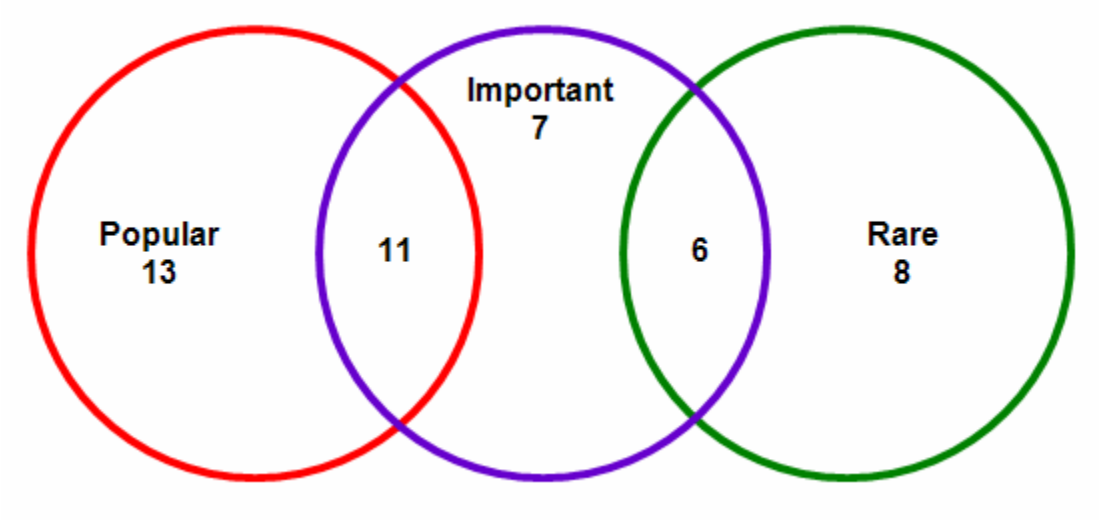


Figure 54. Venn diagram of important, popular and rare features.

7.3.2 Discussion

The control group participants were able to predict many popular features; however, there were still many popular features they did not identify. If the designers had been given more slots to identify popular features, they might have done better with this prediction. Regardless, their identification of the most popular was still off. This is an expected outcome. Getting at knowledge that end users possess but that software engineers do not know is one of the motivations for including users in the software

design process to. This is true even when the designers are part of the end user population as discussed earlier.

One premise for including a large number of end users is to generate a more complete set of information for further expansion, analysis, and refinement. Since designers were identifying higher level features and could summarize across end users, the second study was not able to distinguish the types of information contained in end user expressions. In any case, the data indicates the range of data that can be obtained from a large group of people regardless of the mode of expression.

As pointed out before, rare items might be extremely important and desired by other members of the end user population if they were aware of it. So, when presented with these concepts other end users might see these as being necessary. In this study, designers identified features that were rare in the set of information they were exploring. A rare feature not included as an important feature could well be a feature that might become crucial. This approach provides a way of collecting rare features that could be an important part of a software design effort. These features would remain unidentified if using a small sample of representative users.

7.4 DE Tool and Process

The questions in the follow up survey (Table VIII and Table IX) serve two purposes. The first is to determine participants' perceptions of the process of collecting user expression from a set of users via the mode of expression used in their condition and using that expression to identify domain and software requirements information. The second is to gain insight into how the tool assists and hinders this process. Such an

understanding can be used to improve use of this process and the iterative development of tools that support it.

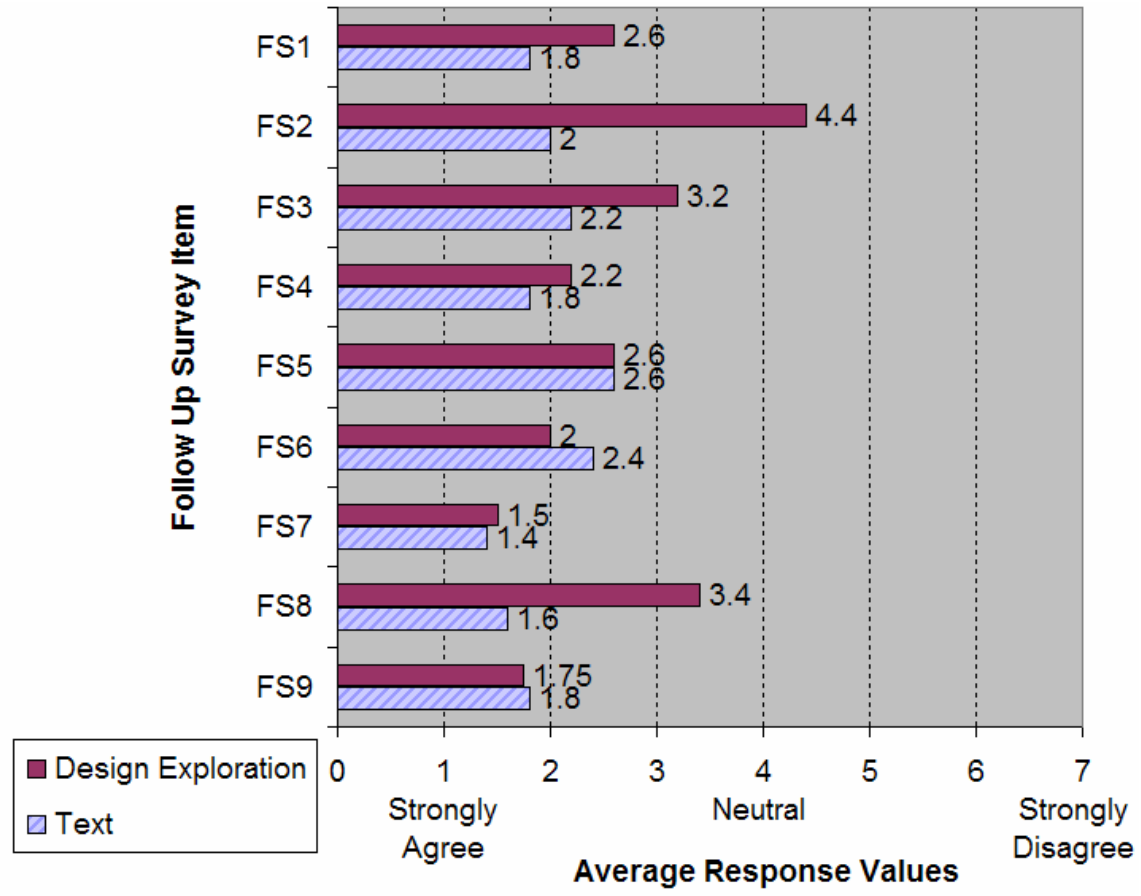


Figure 55. Average response values for follow up survey.

7.4.1 Quantitative Responses

In the first set of scaled responses, questions FS1, FS3, FS4, and FS9 address the process. FS1 provides a rough quality assessment via designer's perceptions of the domain information they identified. FS3 and FS4 probe how well designers saw their understanding of end user expression. FS9 asks their opinion on the viability of the

process for collecting information from users. The average response from all designers fell on the agreement side of neutral (Figure 55). However, in all cases except FS9 agreement was stronger for designers in the text condition than those in the Design Exploration condition.

Questions FS2, FS5, FS6, FS7 and FS8 address the use of a tool for the process. FS2 and FS8 deal specifically with the tool used in the study. FS2 deals with navigating user expression with the tool provided. The FS2 response for the Design Exploration condition (4.4) fell on the disagree side of neutral (4) whereas the response for the text condition was firmly on the agree side (2) (Figure 55). FS8 directly asks whether the tool assisted the process. While both responses were affirmative, the response for the Design Exploration group was barely to the agree side of neutral.

FS5, FS6 and FS7 deal with the general idea of whether a tool is needed for the process. FS7 directly asks whether a tool is needed to assist the process. FS5 and FS6 seek this information more indirectly by asking whether having more user expressions to analyze would significantly increase difficulty and time respectively. Users in both conditions agreed that it would.

The last four questions ask about the helpfulness of specific features found in the tool they used. All four were given to designers in the Design Exploration condition. Only the first two were given to designers in the text condition because of their modified tool. Both groups found search overlay and the terms helpful (Figure 56). Similarity navigation received a neutral response, and clusters received a response just to the helpful side of neutral.

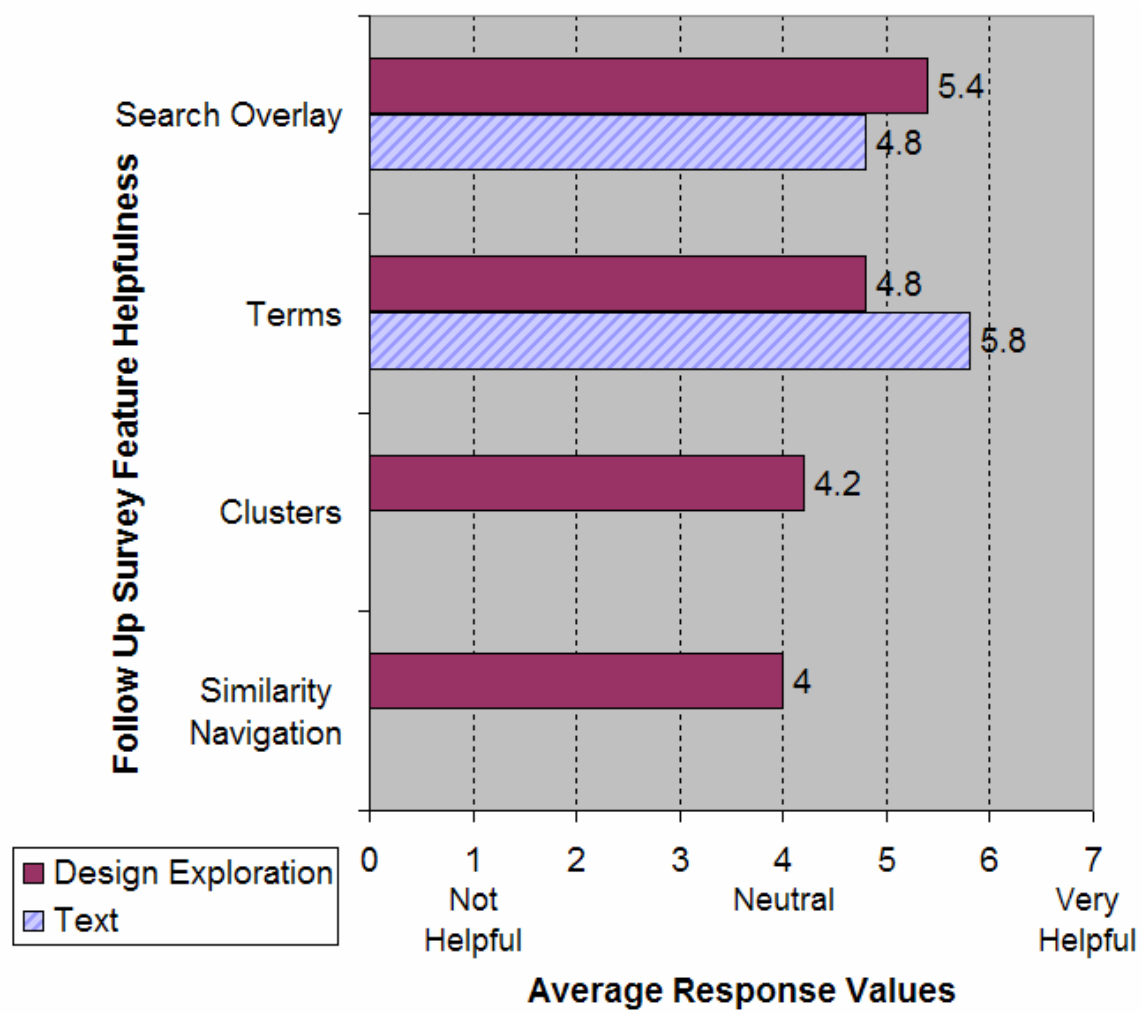


Figure 56. Average responses for feature helpfulness.

7.5 Summary

The study of Design Exploration Analyzer provided insight into information garnered through the Design Exploration process in relation to information gathered textually and information gathered without end user input. The amenities and features identified show that there were large areas where information gathered in each condition did not overlap indicating that the information gathered in each approach was different.

However, the study was not designed to assess differences in the types of information produced by end users in each condition.

While not overwhelmingly positive, designers saw value in the information gathered in each condition. Results might have been influenced by the time constraints imposed in the study since information provided by end users was missed by designers exploring the information space. Regardless, designers were able to use the tools supplied to assist their navigation and analysis. Further tool improvements should improve the ability to explore and analyze end user expressions.

Overall the approach is good. It gets needed information from a large set of end users expressions; however, this type of analysis is a new activity for software designers and needs support. The current use of textual and spatial analysis for clustering does provide navigational support, but is not sufficient, especially for designers new to the process. As with most techniques for gathering information from users, software designers must learn to use and understand the data collected. The evaluation occurred in a context where designers did not have enough time to go through the learning process needed to understand and utilize this approach and end user expressions.

8. OPEN ISSUES AND FUTURE WORK

This dissertation has only scratched the surface of exploring the application of this approach to collecting and analyzing end users expressions for software design. The current system uses separate Java applications to collect and analyze end user expressions. However, as pointed out earlier, end users can be geographically distributed. Moreover, with all of the security issues prevalent in current computing environments, users will be unwilling to install an application onto their computer. Additionally, users would then need to send the files generated containing their partial designs to software designers. Many of these logistic issues can be resolved by implementing this approach via a web-based framework as well as enable additional interactions. Current work is converting the Design Exploration Builder and Analyzer for use within a web-base framework.

End users indicated some frustration with the limitations of their modes of expression. In some cases the limitations imposed were inadvertent. Future versions of the Design Exploration Builder should look at ways to expand the modes of expression available for users. Future versions should allow selection of different modes of expression including textual expression. In many instances, the desired expressiveness is related to aesthetic modifiability. However, this could confound data collection. A challenge is to provide extended expressiveness while keeping end user focus on semantic content.

One type of expressiveness desired by some users was to specify explicit relationships between widgets and other windows, e.g. showing the window that a button

opens. However, adding this type of functionality to the Design Exploration Builder adds additional complexity to the application. Consequently, this needs to be done in a way that maintains the sit down and use nature of this application. Collection of formal information from end users will enhance the automatic analysis that supports exploration by designers. For example, the spatial parser can be imprecise. Getting users to create interface structures recognized by the spatial parser would enhance analysis. Finding ways to encourage the expression of formal information without hindering end users' free form expressions remains an important area for research.

In Design Exploration's current incarnation, end users work in isolation while creating their designs. A next step is exploring how collaboration in various forms would influence the process. This collaboration could take several forms whether synchronous or asynchronous. For example, commenting systems are common and this type of approach could be a way for users and designers to provide feedback and get clarification for different aspects of partial designs.

A branching history mechanism is built into the Design Exploration Builder. However, since its existence was hidden from end users, it was not utilized. Making history visible to end users might encourage the exploration of different design paths. Regardless, history can provide insight into the constructive process. This temporal aspect of user expression needs to be examined and understood. One further use of history could occur in the context of collaboration. If users are aware of each others histories, they might use points in those histories as starting points for their own

contributions. One challenge is to allow this cross seeding while avoiding design fixation [Jansson and Smith 1991].

Sketching interfaces provide many affordances for collecting partial designs that maintains the focus on semantics and away from aesthetic tuning. The current version of the Builder does not support sketching since most users do not have a pen based apparatus that enables sketching and, sketching is difficult using a mouse. However, systems with a pen based interface that can support sketching are more common than in the past. This mode for collecting user input should be revisited. A system that provides both the current approach for interface design as well as a sketching option would provide a system that works for all users and allow sketching for those so equipped. While research as been done in this area that focused on designer sketching [Landay 1996; Landay and Myers 2001], sketching by end users probably has different characteristics that will need to be understood to integrate it into the Design Exploration process.

As stated previously, support is needed to assist designers as they analyze a large number of partial designs. This support takes two forms: access and analysis. The current version of the Design Exploration Analyzer provides access. However, there is no support for analysis. In most cases when working with a few users, a more comprehensive analysis of each representative user's data is performed. However, in the Design Exploration process it is impractical to do this type of in depth analysis of each user's partial design. This is similar to users being overwhelmed with information from internet searches. What is needed is something to help designers as they sort through and

prioritize the various partial designs provided by users since it will be impractical to look at all partial designs in depth. This is similar to the task of document triage but for partial designs instead of documents [Marshall and Shipman 1997]. The unique characteristics of triage for software design needs to be understood and supported.

9. CONCLUSIONS

End user involvement in software design is considered crucial. There are a variety of methods for including users in design. Methods such as surveys and questionnaires can gather data from a large population. But, the formatting of questions to assist the process of analysis constrains the expression of information. Consequently, the richness of the information is limited. Richer information can be garnered from users with face to face methods, but require more effort per user and so only a few representative users can be involved. This dissertation presented Design Exploration, an approach that falls somewhere between these two extremes.

In Design Exploration, end users communicate their desires for what the software should do by creating interface mock ups augmented with textual annotations. Interface construction provides an engaging environment that situates users into the context of the application and its use. The partial designs created by a set of end users are collected and analyzed by software designers.

Two tools were developed to support this process, the Design Exploration Builder and Analyzer. The Design Exploration Builder supports end user generation of low fidelity partial designs via end user creation of windows and the arrangement of widgets in them. Textual descriptions can also be added to express information that might not be obvious or that is difficult to express using only the visual representation. The Analyzer tool supports the navigation and search of annotated partial designs using a combination of both textual and spatial analysis.

The Builder study compared user expressions generated textually, through interface construction without the ability to annotate, and through interface design with annotation capability. Only the text group felt they expressed everything they wanted to express; however, people are probably more skilled at expressing themselves textually since this is something that is taught in school and is more practiced.

Regardless, the affordances of different modes of communication were desired. Those in the textual condition felt it would be good to be able to show layout information and those in the interface construction conditions wanted to be able to make more global textual comments. In some cases, users bypassed the constraints imposed in the study to express information. These results indicate that user expression in this process should not be constrained, but enabled in such a way that users can choose their mode of expression whether it is interface construction, textual descriptions or both. They can then choose the mode that is most comfortable to them or is most conducive for communicating particular information. Moreover, users in the interface construction conditions expressed details and types of information that were not generated in the text only condition.

The analyzer study compared information identified by software designers that was gathered in three conditions. In the control condition, software designers generated responses without the benefit of user expression. In the two other conditions software designers generated responses by mining end user expressions gathered through either the Design Exploration process or textually. Moreover, designers' perceptions of the Design Exploration process were determined.

End users liked and disliked expressing information in their various conditions. It is likely that some of these dislikes were due to constraints hindering preferred modes of expression. In fact some subjects in the first study did not want to stop when their time was completed. Communicating in a user's preferred mode of expression results in a more engaging activity. So freedom of expression coupled with the extended amount of information available from a large population allows Design Exploration to be a process that collects information from a large number of users in a more engaging manner than questionnaires.

The Analyzer study showed that the tasks of identifying amenities and software features produced approximately equivalent information in each condition. While the study was not able to distinguish between the types of information and the details contained in the information collected textually and through interface construction, the large amount of unique concepts identified show that different information was garnered from each mode of user expression.

The evaluation shows that Design Exploration is a viable approach to collecting and analyzing information from a large number of users in spite of deficits in the tool design. End user expression of domain and software information occurs through the Builder tool and analysis of end users expressions by software designers occurs through the Analyzer tool. Further work is necessary on supporting the analysis of collections of annotated partial designs. Regardless, the Design Exploration process adds a technique to software designers' toolbox that fills a void for engaging a larger number of end users

in software design that does not loose all of the richness of more in depth face to face approaches.

REFERENCES

BEYER, H. AND HOLTZBLATT K. 1999. Contextual design, *interactions 6*, 1, 32-42.

BØDKER, S. 2000. Scenarios in user-centred design - setting the stage for reflection and action. *Interacting with Computers 13*, 61-75.

BOLAND, J.R. 1978. The Process and Product of System Design, *Management Science 24*, 9, 887-898.

CARROLL, J.M. 2000. Five reasons for scenario-based design. *Interacting with Computers 13*, 43-60.

CARROLL J.M., ROSSON, M.B., CHIN, G. AND KOENEMANN, J. 1997. Requirements Development: Stages of opportunity for collaboration needs discovery, In *Proceedings of the Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*. Amsterdam, The Netherlands, Aug. 55-64.

CHIN, G., ROSSON AND M.B., CARROLL, J.M. 1997. Participatory analysis: Shared development of requirements from scenarios, human factors in computing systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Atlanta, GA, Mar. 162-169.

COOPER, A. 2004. *The Inmates are Running the Asylum*. Pearson Education, Indianapolis, IN.

CURTIS, B., KRASNER, H., AND ISCOE, N. 1988. A field study of the software design process for large systems. *Communications of the ACM 31*, 11, 1268-1287.

CYPHER, A. 1993. Introduction. In *Watch What I Do: Programming by Demonstration*, Cypher, A. Ed. MIT Press, Cambridge, MA 1-11.

CYPHER, A., KOSBIE, D.S., AND MAULSBY, D. 1993. Characterizing PBD Systems. In *Watch What I Do: Programming by Demonstration*, Cypher, A. Ed. MIT Press, Cambridge, MA 467-484.

DARDENNE, A., FICKAS, S., AND VAN LAMSWEERDE, A. 1991. Goal-directed concept acquisition in requirements elicitation. In *Proceedings of 6th International Workshop on Software Specification and Design.* Como, Italy, Oct. 14-21.

DERTHICK, M. AND ROTH, S. F. 2001. Example based generation of custom data analysis appliances. In *Proceedings of the 6th International Conference on Intelligent User Interfaces*. Santa Fe, NM, Jan. 57-64.

EHN, P. 1988. Playing the language-games of design and use-on skill and participation. In *Proceedings of the ACM SIGOIS and IEEECS TC-OA 1988 Conference on Office Information Systems*. Palo Alto, CA, Mar. 142-157.

EHN, P. 1993. Scandinavian design: On participation and skill. In *Participatory Design: Principles and Practices*. Schuler, D. and Namioka, A. Eds. Lawrence Earlbaum Associates, Hillsdale, NJ 41-77.

FARRELL, V., FARRELL, G., MOUZAKIS, K., PILGRIM, C., AND BYRT, P. 2006. PICTIOL: A case study in participatory design. In *Proceedings of the 20th Conference of the Computer-Human Interaction Special Interest Group (CHISIG) of Australia on Computer-Human Interaction: Design: Activities, Artefacts and Environments.* Sydney, Australia, Nov. 191-198.

FINKELSTEIN, A. 1994. Requirements engineering: A review and research agenda. In *Proceedings of 1st Asian-Pacific Software Engineering Conference*. Tokyo, Japan, Dec. 10-19.

GLENBERG, A.M. AND MCDANIEL, M.A. 1992. Mental models, pictures, and text: Integration of spatial and verbal information. *Memory and Cognition 20,* 5, 458-460.

GOGUEN, J.A. AND LINDE, C. 1993. Techniques for requirements elicitation. In *Proceedings, Requirements Engineering '93*. San Diego, CA, Jan. 152-164.

GRUDIN, J. AND PRUITT J. 2002. Personas, participatory design and product development: An infrastructure for engagement. In *Proceedings of the 7th Biennial Participatory Design Conference*. Malmö, Sweden, June 144-161.

HOLBROOK, H. 1990. A scenario-based methodology for conducting requirements elicitation. *SIGSOFT Software Engineering Notes 15*, 1, 95-104.

HOLTZBLATT, K. AND BEYER, H.R. 1995. Requirements gathering: The human factor. *Communications of the ACM 38*, 5, 31-32.

HOLTZBLATT, K. AND BEYER, H. 2003. A tool supporting capture and analysis of field research data using the contextual design methodology. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems*. Ft. Lauderdale, FL, Apr. 630-631.

JANSSON, D. G. AND SMITH, S. M. 1991. Design fixation. *Design Studies 12*, 1, 3-11.

KYNG, M. 1995. Creating context for design. In *Scenario-Based Design: Envisioning Work and Technology in System Development*, Carroll, J.M. Ed. J. Wiley, New York 85-107.

LANDAY, J.A. 1996. SILK: Sketching interfaces like krazy. *In Conference Companion on Human Factors in Computing Systems: Common Ground*. Vancouver, British Columbia, Canada, Apr. 398-399.

LANDAY, J.A. AND MYERS, B.A. 1995. Interactive sketching for the early stages of user interface design, In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems.* Denver, CO, May, 43-50.

LANDAY, J.A. AND MYERS, B.A. 2001. Sketching interfaces: Toward more human interface design. *Computer 34*, 3, 56-64.

MARSHALL, C.C. AND SHIPMAN, F.M. 1997. Spatial hypertext and the practice of information triage. In *Proceedings of the Eighth ACM Conference on Hypertext*. Shouthhampton, United Kingdom, Apr. 124-133.

MILLER, D. S., SMITH, J. G., AND MULLER, M. J. 1992. TelePICTIVE: Computer-supported collaborative GUI design for designers with diverse expertise. In *Proceedings of the 5th Annual ACM Symposium on User interface Software and Technology*. Monteray, CA, Nov. 151-160.

MOORE, J.M. 2003. Communicating requirements using end-user GUI constructions with argumentation. In *Proceedings 18th IEEE International Conference on Automated Software Engineering*. Montreal, Canada, Oct. 360-363.

MOORE, J.M. AND SHIPMAN, F.S. 2000. A comparison of questionnaire-based and GUI-based requirements gathering, In *Proceedings of the 15th IEEE International Conference on Automated Software Engineering*, Grenoble, France, Sept. 35-43.

MOORE, J.M. AND SHIPMAN, F.S. 2001. Requirements elicitation using visual and textual information, In *Proceedings of the, 5th International Symposium on Requirements Engineering*. Toronto, Canada, Aug. 308-309.

MULLER, M.J. 1991. PICTIVE—an exploration in participatory design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Reaching Through Technology*. New Orleans, LA, Apr. 225-231.

MULLER, M.J. 2002. Participatory design: The third space in HCI. In *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications,* Jacko, J.A. and Sears, A. Eds. Lawrence Erlbaum Associates, Mahwah, NJ 1051-1068.

MULLER, M.J., WILDMAN, D.M., AND WHITE, E.A. 1993. 'Equal opportunity' PD using PICTIVE. *Communications of the ACM 36*, 4. 64-66.

NIELSEN, J., CLEMMENSEN, T., AND YSSING, C. 2002. Getting access to what goes on in people's heads?: Reflections on the think-aloud technique. In *Proceedings of the Second Nordic Conference on Human-Computer Interaction.* Aarhus, Denmark, Oct. 101-110.

NUSEIBEH, B. AND EASTERBROOK, S. 2000. Software engineering: A roadmap. In *Proceedings of the Conference on the Future of Software Engineering.* Limerick, Ireland, June 3-22.

POLANYI, M. 1966. *The Tacit Dimension*. Doubleday. Garden City, NY.

POTTS, C. 1999. ScenIC: A strategy for inquiry-driven requirements determination. In *Proceedings of the 4th IEEE International Symposium on Requirements Engineering*. Limerick, Ireland, June 58-65.

POTTS, C., TAKAHASHI, K., AND ANTON, A.I. 1994. Inquiry-based requirements analysis. *IEEE Software 11*, 2, 21-32.

REEVES, B. 1993. Supporting collaborative design by embedding communication and history in design artifacts. PhD. Thesis. Department of Computer Science, University of Colorado at Boulder.

REEVES, B. AND SHIPMAN, F. 1992. Supporting communication between designers with artifact-centered evolving information spaces, In *Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work*. Toronto, Canada, Nov. 394-401.

REUBENSTEIN, H.B. AND WATERS, R.C. 1991. The requirements apprentice: Automated assistance for requirements acquisition. *IEEE Transactions on Software Engineering 17*, 3. 226-240.

RUDD, J., STERN, K., AND ISENSEE, S. 1996. Low vs. high-fidelity prototyping debate. *interactions 3*, 1, 76-85.

SCHULER, D. AND NAMIOKA, A. 1993. Preface. In *Participatory Design: Principles and Practices*, Schuler, D. and Namioka, A. Eds. Lawrence Earlbaum Associates, Hillsdale, NJ, xi-xiii.

SHAH, J.J., VARGAS-HERNANDEZ, N., AND SMITH, S.M. 2003. Metrics for measuring ideation effectiveness. *Design Studies 24*, 111-134.

SHIPMAN, F.M. AND HSIEH, H. 2000. Navigable history: A reader's view of writer's time, *New Review of Hypermedia and Multimedia 6*, 147-167.

SHIPMAN, F.M., HSIEH, H., AIRHART, R., MALOOR, P., AND MOORE, J.M. 2001a. The visual knowledge builder: A second generation spatial hypertext. In *Proceedings of the Twelfth ACM Conference on Hypertext and Hypermedia*. Århus, Denmark, Aug. 113-122.

SHIPMAN, F.M., HSIEH, H., AIRHART, R., MALOOR, P., MOORE, J.M., AND SHAH, D. 2001b. Emergent structure in analytic workspaces: Design and use of the visual knowledge builder. In *Proceedings of IFIP INTERACT'01: Human-Computer Interaction*. Tokyo, Japan, July 132-139.

SHIPMAN, F.M. AND MARSHALL, C.C. 1999. Formality considered harmful: Experiences, emerging themes, and directions on the use of formal representations in interactive systems. *Computer Supported Cooperative Work (CSCW) 8*, 4, 333-352.

SHIPMAN, F. M., MARSHALL, C. C., AND LEMERE, M. 1999. Beyond location: Hypertext workspaces and non-linear views. In *Proceedings of the Tenth ACM Conference on Hypertext and Hypermedia*. Darmstadt, Germany, Feb. 121-130.

SHIPMAN, F. M., MARSHALL, C. C., AND MORAN, T. P. 1995. Finding and using implicit structure in human organized spatial layouts of information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Denver, Colorado, May 346-353.

SHIPMAN, F. M. AND MCCALL, R. 1994. Supporting knowledge-base evolution with incremental formalization. In *Conference Companion on Human Factors in Computing Systems*. Boston, MA, Apr. 285-291.

SUCHMAN, L. 1987. *Plans and Situated Actions*. Cambridge University Press, Cambridge, UK.

SUCHMAN, L. AND JORDAN, B. 1990. Interactional troubles in face-to-face survey interviews. *Journal of the American Statistical Association 85*, 409, 232-241.

SUTCLIFFE, A. 1995. Requirements rationales: Integrating approaches to requirements analysis. In *Proceedings of the Conference on Designing interactive Systems: Processes, Practices, Methods, & Techniques*. Ann Arbor, MI, Aug. 33-42.

VAN LAMSWEERDE, A. 2000. Requirements engineering in the year 00: A research perspective. *In ICSE '00: Proceedings of the 22nd International Conference on Software Engineering*. Limerick, Ireland, June 5-19.

VILLER, S. AND SOMMERVILLE, I. 1999. Social analysis in the requirements engineering process: From ethnography to method. In *Proceedings of the 4th IEEE International Symposium on Requirements Engineering*. Limerick, Ireland, June 6-13.

WILSON, S. AND JOHNSON, P. 1995. Empowering users in a task-based approach to design. In *Proceedings of the Conference on Designing interactive Systems: Processes, Practices, Methods, & Techniques*. Ann Arbor, MI, Aug. 25-31.

WITTEN, I. 1999. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers. San Francisco, CA.

ZAVE, P. AND JACKSON, M. 1997. Four dark corners of requirements engineering. *ACM Transactions on Software Engineering and Methodology 6*, 1, 1-30.

APPENDIX A

DESIGN EXPLORATION STOPWORDS

1. about
2. access
3. accesses
4. all
5. area
6. back
7. box
8. box
9. button
10. buttons
11. cancel
12. check
13. click
14. clicked
15. clicking
16. close
17. combo
18. continue
19. data
20. database
21. downclick
22. rightclick
23. enter
24. entered
25. exit
26. field
27. finish
28. finished
29. go
30. input
31. label
32. labeled
33. link
34. list
35. main
36. menu
37. ok
38. open
39. opened
40. opening
41. opens
42. option
43. options
44. page
45. print
46. press
47. pressed
48. push
49. pushed
50. quit
51. radio
52. screen
53. screens
54. select
55. selected
56. selection
57. selects
58. showing
59. submit
60. take
61. takes
62. text
63. their
64. theirs
65. thing
66. things
67. title
68. titled
69. user
70. widget
71. widgets
72. window
73. windows
74. a
75. b
76. c
77. d
78. e
79. f
80. g
81. h
82. i
83. j
84. k
85. l
86. m
87. n
88. o
89. p
90. q
91. r
92. s
93. t
94. u
95. v
96. w
97. x
98. y
99. z

APPENDIX B

ANALYZER STUDY TUTORIALS

Tutorials were provided to study participants using a version of the Design Exploration Analyzer. The web pages for the Design Exploration tutorial (tutorialDE.zip) and the Text tutorial (tutorialText.zip) are located in zipped folders that can be found with the files that accompany this dissertation.

APPENDIX C

ANALYZER STUDY SCENARIOS

- Design Exploration

You are working for a software development company, and are preparing information regarding a new project for your supervisor.

The project is to develop software that college students will use to assist them in their search for housing. The company is using the Design Exploration process (See web information) to solicit initial information from a set of potential users. They were given the following task:

> A software company is developing a software program to help college students find housing. Since college students will be using this program, you have been asked to provide input for the program.
>
> You are to provide as much information about what this program should do during the next hour. Please start with what you think is most important.

Each user provided this information using the Design Exploration creation tool to create partial designs.

Prior to writing formal requirements, you are trying to get a grasp of the domain and user expectations. For your report, you are to provide the following information.

Note: When asked for rationale, you should support your opinion. If the support is based on user data, you should refer to that information (e.g. partial designs, windows, widgets, exploration tool features) so others can locate those information sources.

Note: Rationale should be succinct since time is limited.

- Text

You are working for a software development company, and are preparing information regarding a new project for your supervisor.

The project is to develop software that college students will use to assist them in their search for housing. They were given the following task:

> A software company is developing a software program to help college students find housing. Since college students will be using this program, you have been asked to provide input for the program.
>
> You are to provide as much information about what this program should do during the next hour. Please start with what you think is most important.

Each user provided this information typing textual descriptions into MS Word..

Prior to writing formal requirements, you are trying to get a grasp of the domain and user expectations. For your report, you are to provide the following information.

Note: When asked for rationale, you should support your opinion. If the support is based on user data, you should refer to that information (e.g. documents, exploration tool features) so others can locate those information sources.

Note: Rationale should be succinct since time is limited.

- Control

You are working for a software development company, and are preparing information regarding a new project for your supervisor.

The project is to develop software that college students will use to assist them in their search for housing.

Prior to writing formal requirements, you are trying to get a grasp of the domain and user expectations. For your report, you are to provide the following information.

Note: When asked for rationale, you should support your opinion.

Note: Rationale should be succinct since time is limited.

APPENDIX D

ANALYZER STUDY DEFINITION PAGE

A Google Definition Page for “amenities” was used to provide a reference definition during the Analyzer study. The definitions page (amenities.htm) that was saved and provided to participants is located in a zipped folder (amenities.zip) that can be found with the files that accompany this dissertation.

APPENDIX E

ANALYZER STUDY AMENITIES GROUPING

The Visual Knowledge Builder (VKB) was used to sort and group the amenities collected in the study into concepts for analysis. The VKB file (amenities.xvkb) can be found with the files that accompany this dissertation. Moreover, the installation package for VKB is included with the files that accompany this dissertation in a zipped folder (VKB_full.zip).

APPENDIX F

ANALYZER STUDY FEATURES GROUPING

The Visual Knowledge Builder (VKB) was used to sort and group the features collected in the study into concepts for analysis. The VKB file (features.xvkb) can be found with the files that accompany this dissertation. Moreover, the installation package for VKB is included with the files that accompany this dissertation in a zipped folder (VKB_full.zip).

VITA

John Michael Moore received his Bachelor of Science degree in biochemistry from Texas A&M University in 1991. He graduated cum laude and with university honors. He received his Master of Science degree in computer science from Southwest Texas State University in 1997. In August 2007 he received his Doctor of Philosophy degree in computer science from Texas A&M University in College Station.

During his doctoral studies, J. Michael Moore lectured introductory programming courses in Pascal and Java for Blinn College in Bryan, Texas and for Texas A&M University in College Station, Texas. He also helped develop and teach an Interaction Design course for Duke University's Talent Identification Program (TIP).

He also worked as a Graduate Research Assistant for the Center for the Study of Digital Libraries. He worked on the Visual Knowledge Builder (VKB) and Design Exploration projects. He published in international conferences such as: IEEE Conference on Automated Software Engineering, ACM conference on Hypertext, IEEE International Symposium on Requirements Engineering, and Human Computer Interaction: INTERACT.

J. Michael Moore can be contacted at:

Texas A&M University
Department of Computer Science
TAMU 3112
College Station, TX 77843-3112

jmichael@cs.tamu.edu